



Bazy wiedzy

Krzysztof Goczyła

Wojciech Waloszek

Teresa Zawadzka

Michał Zawadzki

*Katedra Inżynierii Oprogramowania
Wydział Elektroniki, Telekomunikacji
i Informatyki
Politechnika Gdańsk*





Spis treści

1. Wprowadzenie do ontologii
2. Wczesne ontologiczne metody reprezentacji wiedzy
3. Semantyczny Internet (Semantic Web)
4. Resource Description Framework (RDF)
5. Web Ontology Language (OWL)
6. Logika opisowa (DL) jako fundament ontologicznych metod reprezentacji wiedzy
7. Inżynieria ontologii
8. Źródła

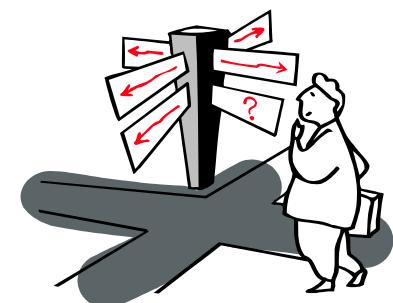




Spis treści...

1. Wprowadzenie do ontologii

2. Wczesne ontologiczne metody reprezentacji wiedzy
3. Semantyczny Internet (Semantic Web)
4. Resource Description Framework (RDF)
5. Web Ontology Language (OWL)
6. Logika opisowa (DL) jako fundament ontologicznych metod reprezentacji wiedzy
7. Inżynieria ontologii
8. Źródła

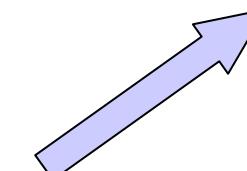
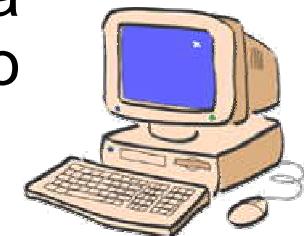




Wiedza

Czym jest wiedza?

- Ogół wiadomości zdobytych dzięki uczeniu się.
- Zbiór informacji zapisanych w pamięci komputera wraz ze zdolnością komputera do samodzielnego poszerzania tego zbioru drogą wnioskowania.



Jak reprezentować wiedzę?



Ontologie

Filozofia:

Ontologia – teoria bytu, podstawowy dział filozofii zajmujący się badaniem charakteru i struktury rzeczywistości.

od greckiego ‘on’ (dopełniacz: ‘ontos’) – ‘byt’, ‘logos’ – ‘nauka’

Informatyka:

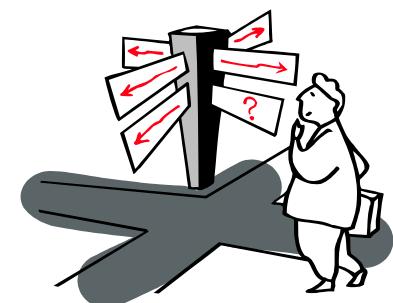
Ontologia - formalny opis pojęć występujących w danej dziedzinie („słownik”, „encyklopedia”).

**Geneza - ramki (ang. *frames*) Minsky'ego (1975);
później: model OKBC (Open Knowledge Base Connectivity)**



Spis treści...

1. Wprowadzenie do ontologii
2. **Wczesne ontologiczne metody reprezentacji wiedzy**
 - **ramki Minsky'ego**
 - **sieci semantyczne**
3. Semantyczny Internet (Semantic Web)
4. Resource Description Framework (RDF)
5. Web Ontology Language (OWL)
6. Logika opisowa (DL) jako fundament ontologicznych metod reprezentacji wiedzy
7. Inżynieria ontologii
8. Źródła





Ontologie wg Minsky'ego

Składniki ontologii wg Minsky'ego:

- **klasy** - pojęcia (*koncepty*)
- **klatki** - właściwości klas opisujące ich cechy i atrybuty
(właściwości, ang. *slot*)
- **fasety** - właściwości klatek (np. ograniczenia nałożone na klatki)
- **aksjomaty** - dodatkowe ograniczenia wyrażone w języku logiki

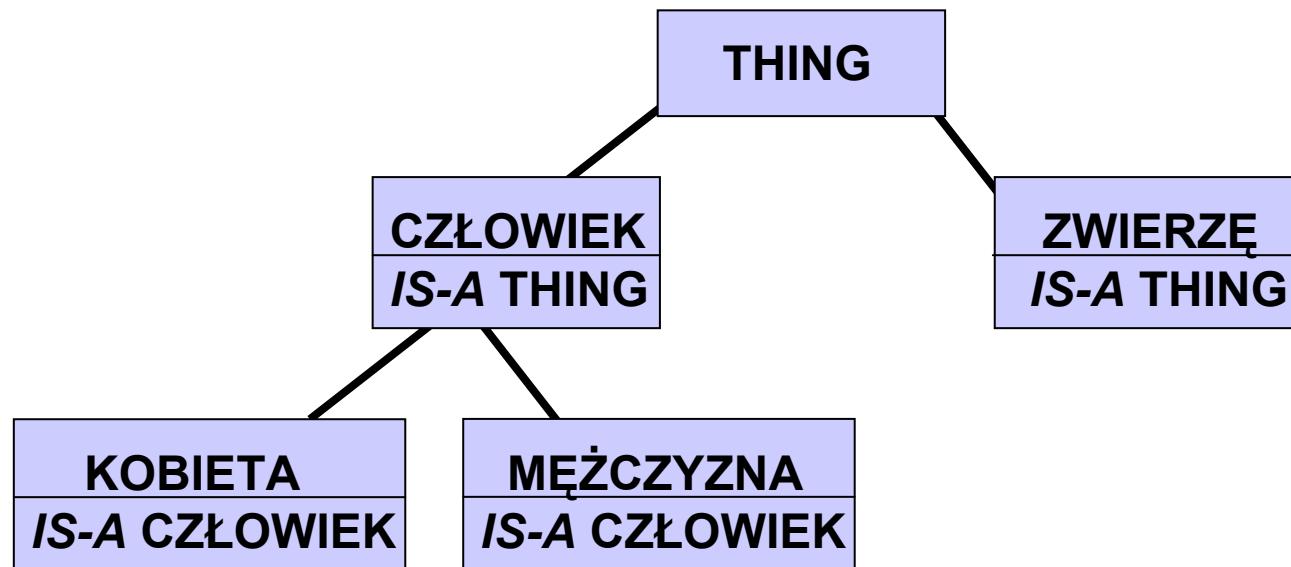
Klasy mają swoje **instancje**, które są wystąpieniami klas z określonymi wartościami klatek.



Klasy

Klasy tworzą hierarchię (taksonomię) klas poprzez relację podklastra - nadklastra.

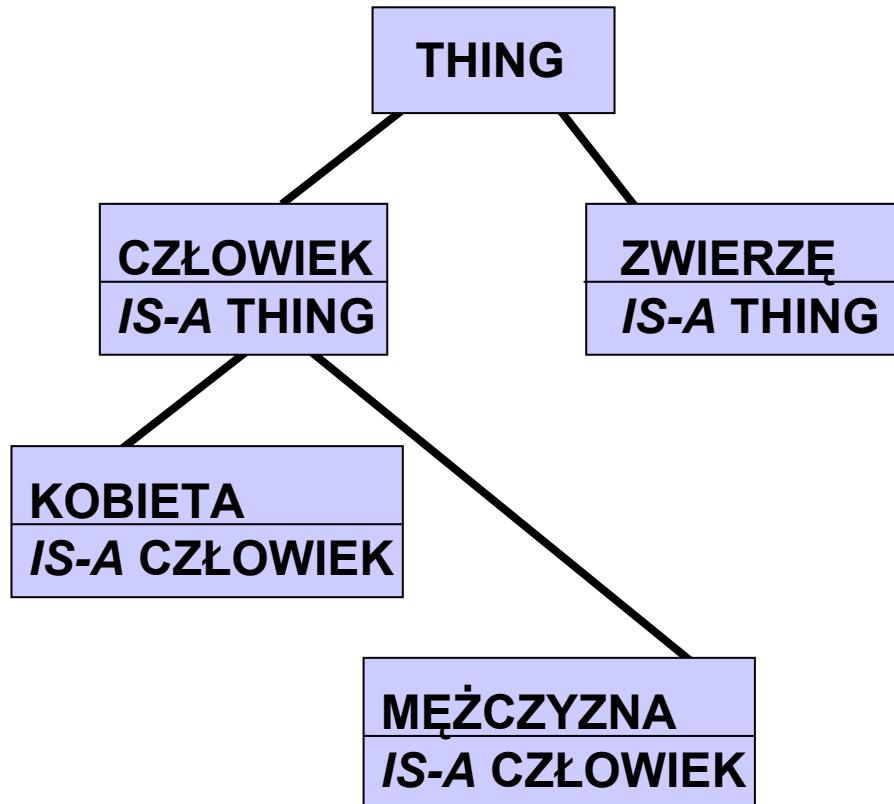
Korzeniem tej hierarchii jest klasa **THING**





Klasy i ich instancje

Klasa A jest podklasą klasy B \Rightarrow każda instancja klasy A jest instancją klasy B



ANNA
INSTANCE_OF KOBIETA



JAN
INSTANCE_OF MĘŻCZYZNA



MAX
INSTANCE_OF ZWIERZĘ





Klatki i fasety

Klatki są definiowane niezależnie od klas (są pełnoprawnymi ramkami).

Klatki są dołączane do klas i – pośrednio – do obiektów.

**Ta sama klatka może być dołączona do wielu klas –
oznacza zawsze to samo.**

Fasety - sposób na nałożenie ograniczenie wartości, jakie mogą przyjmować klatka.

Przykłady faset:

- liczność (liczba różnych wartości)
- wartość minimalna, wartość maksymalna
- dziedzina (zbiór klas, których instancje mogą być wartościami klatki)
- zakres (zbiór klas, do których można dołączyć klatkę)



Klatki i fasety

CZŁOWIEK

IS-A: THING

PŁEĆ: („Ż”, „M”)

ZWIERZĘ

IS-A: THING

PŁEĆ: („Ż”, „M”)

KOBIETA

IS-A CZŁOWIEK

PŁEĆ: „Ż”

MĘŻCZYZNA

IS-A CZŁOWIEK

PŁEĆ: „M”

ANNA

INSTANCE_OF: KOBIETA



JAN

INSTANCE_OF: MĘŻCZYZNA



MAX

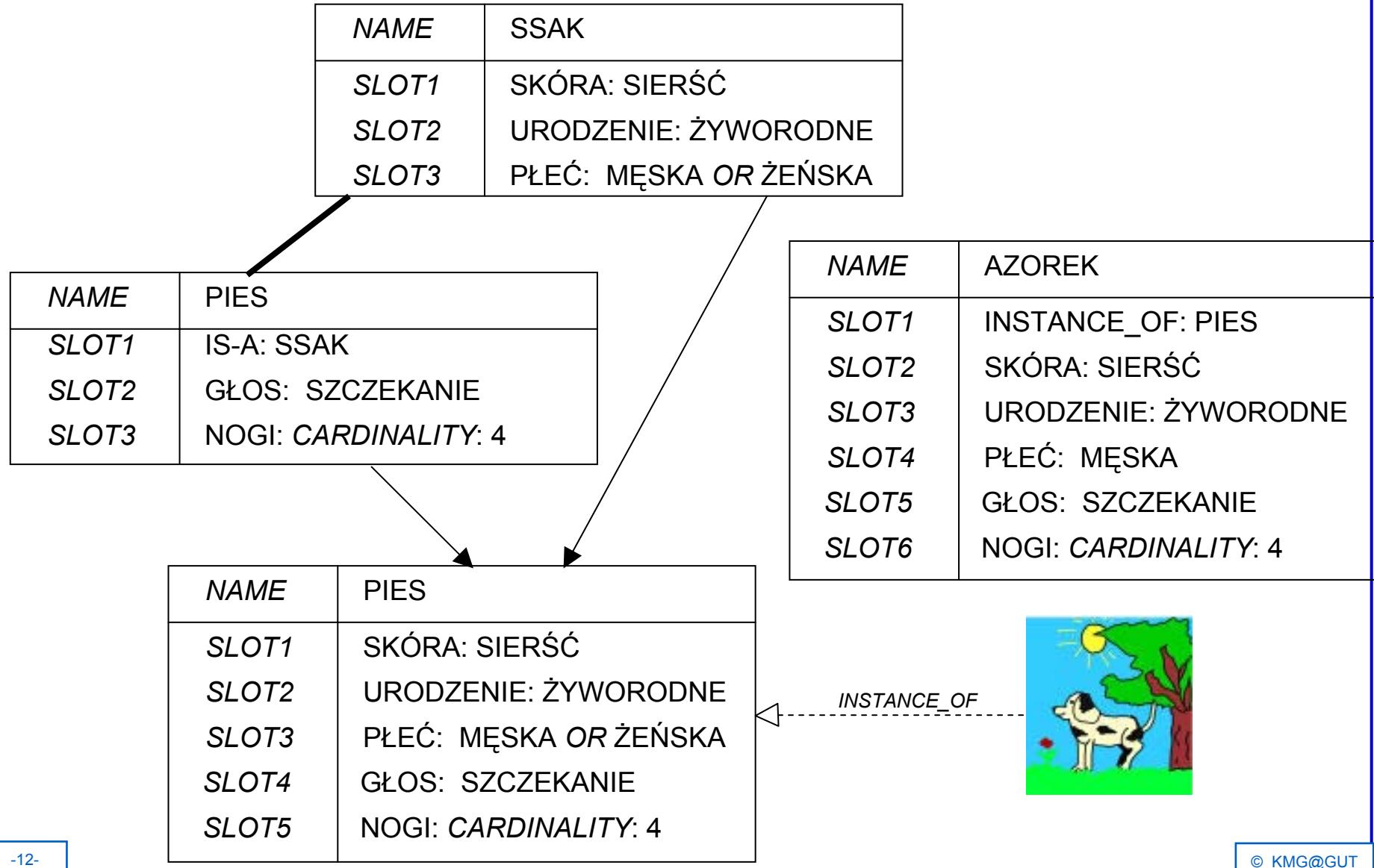
INSTANCE_OF: ZWIERZĘ

PŁEĆ: „M”





Przykład ramek Minsky'ego





Klasy i metaklasy

Klasy tworzą hierarchię (taksonomię) klas poprzez relację podklaśa - nadklaśa.

Korzeniem tej hierarchii jest klasa :THING.

Klasa A jest podklaśią klasy B \Rightarrow każda instancja A jest instancją B

Wystąpienie klasy:



Klasa, której wystąpieniem są klasy, to metaklasa.



Klatki jako ramki

Klatki są definiowane niezależnie od klas (są pełnoprawnymi ramkami).

Klatki są dołączane do klas i – pośrednio – do osobników.

**Ta sama klatka może być dołączona do wielu klas –
oznacza zawsze to samo.**

(UWAGA! Ważna różnica w stosunku do paradygmatu obiektowego!)

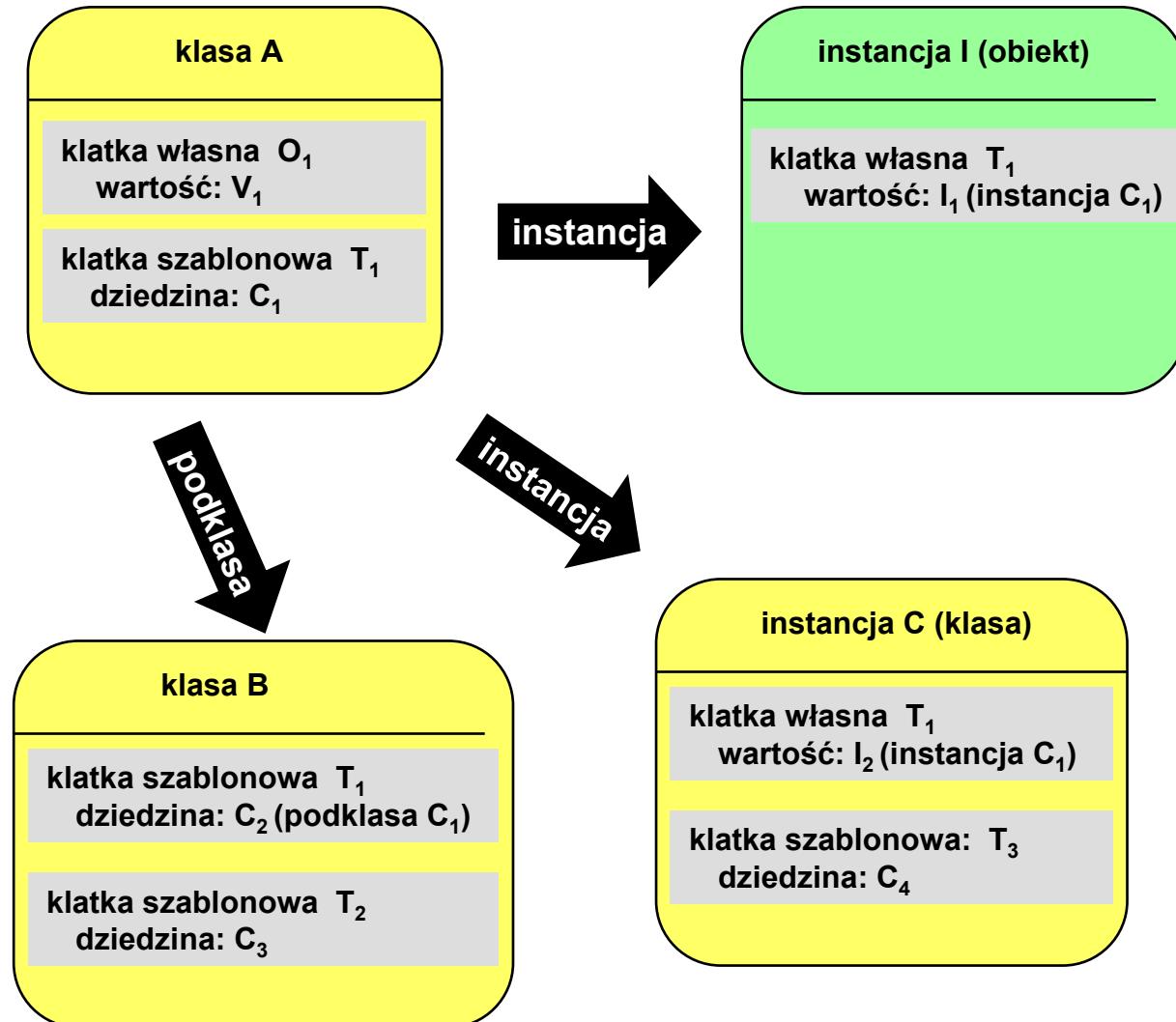
Klatka dołączona do klasy może mieć wartość.

Klatki mogą być *własne* i *szablonowe*.

**Klatki szablonowe mogą być dołączane tylko do klas.
Osobniki mają tylko klatki własne.**

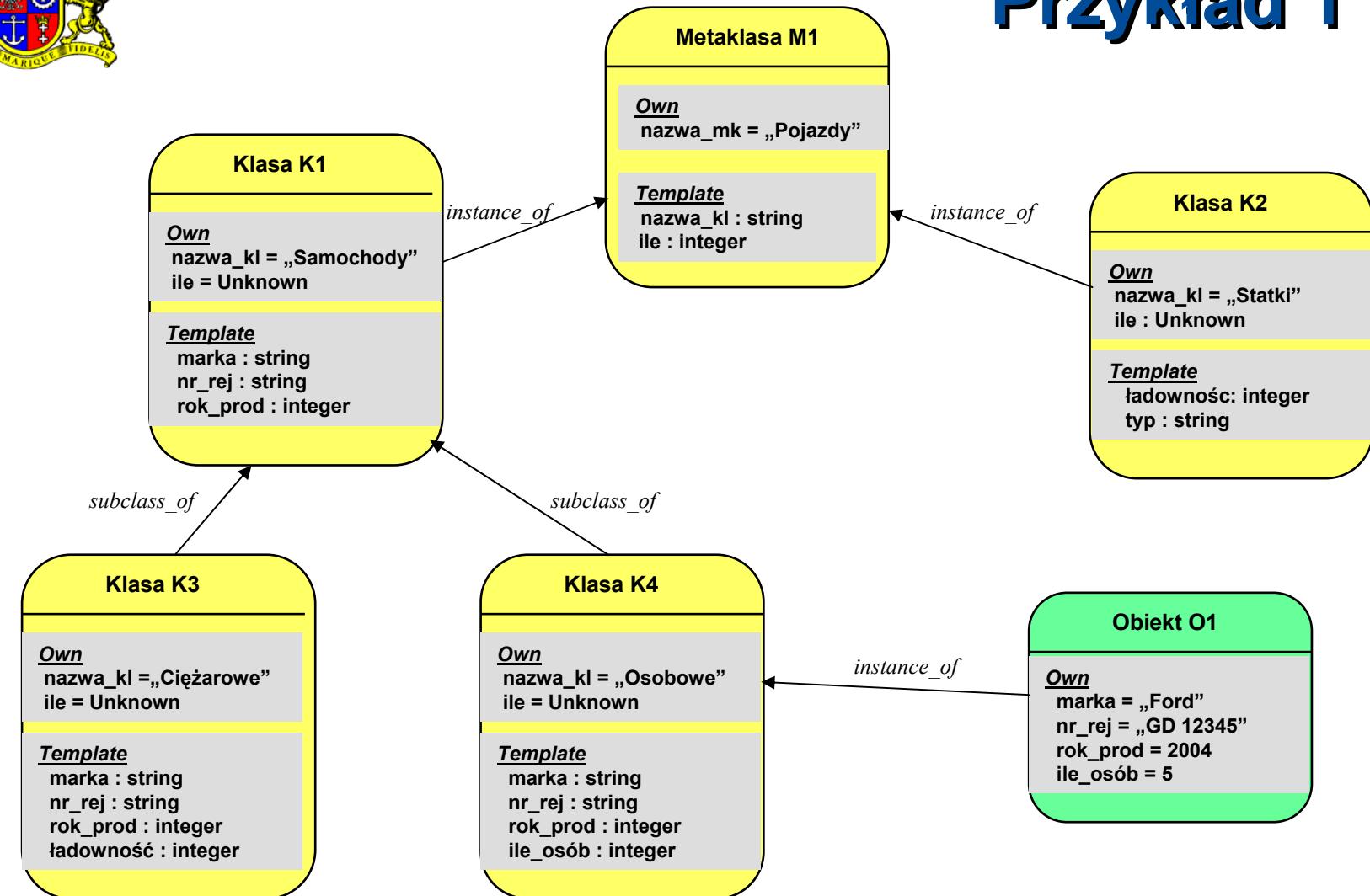


Klatki własne i szablonowe



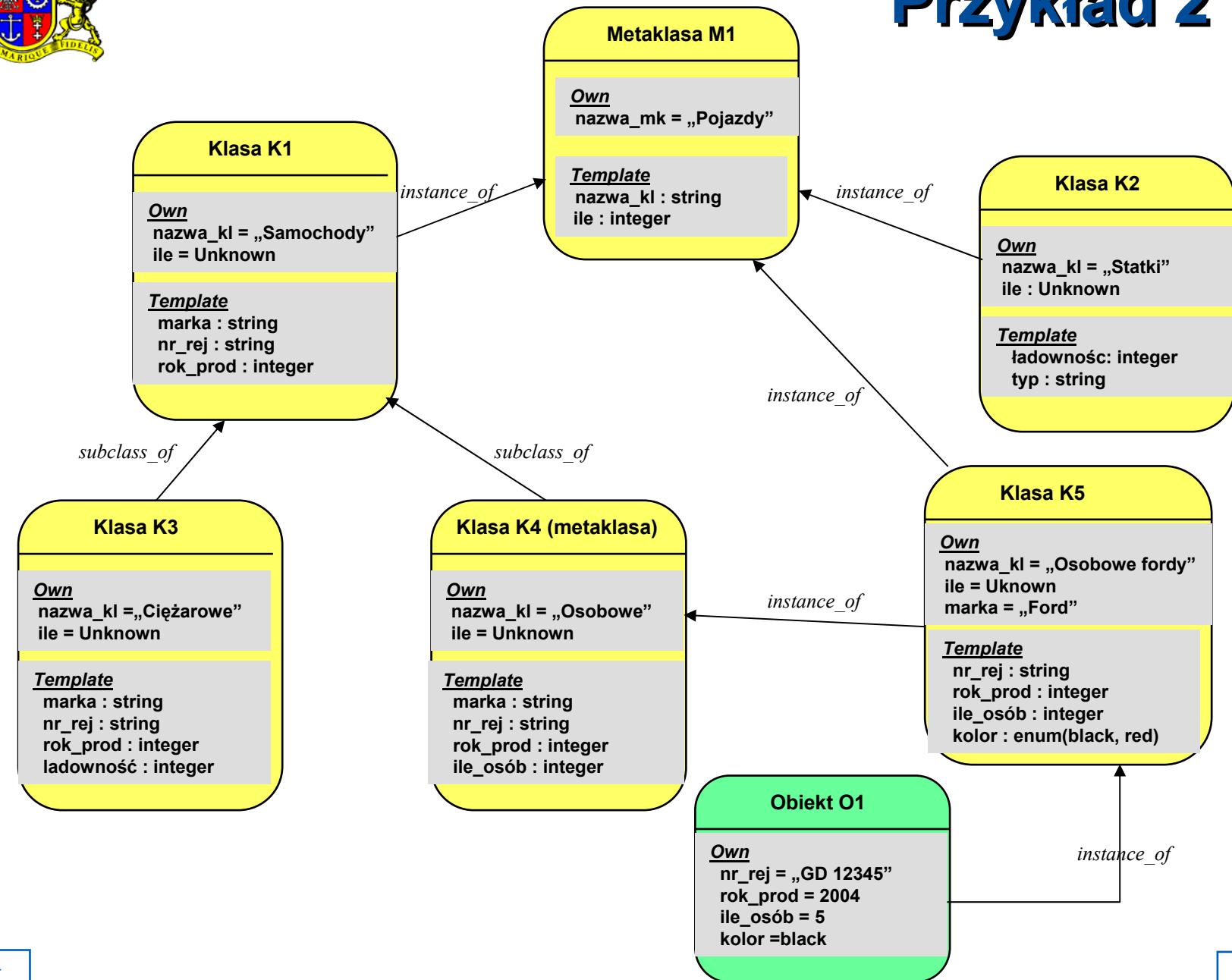


Przykład 1





Przykład 2





Sieć semantyczna

Sieć semantyczna –

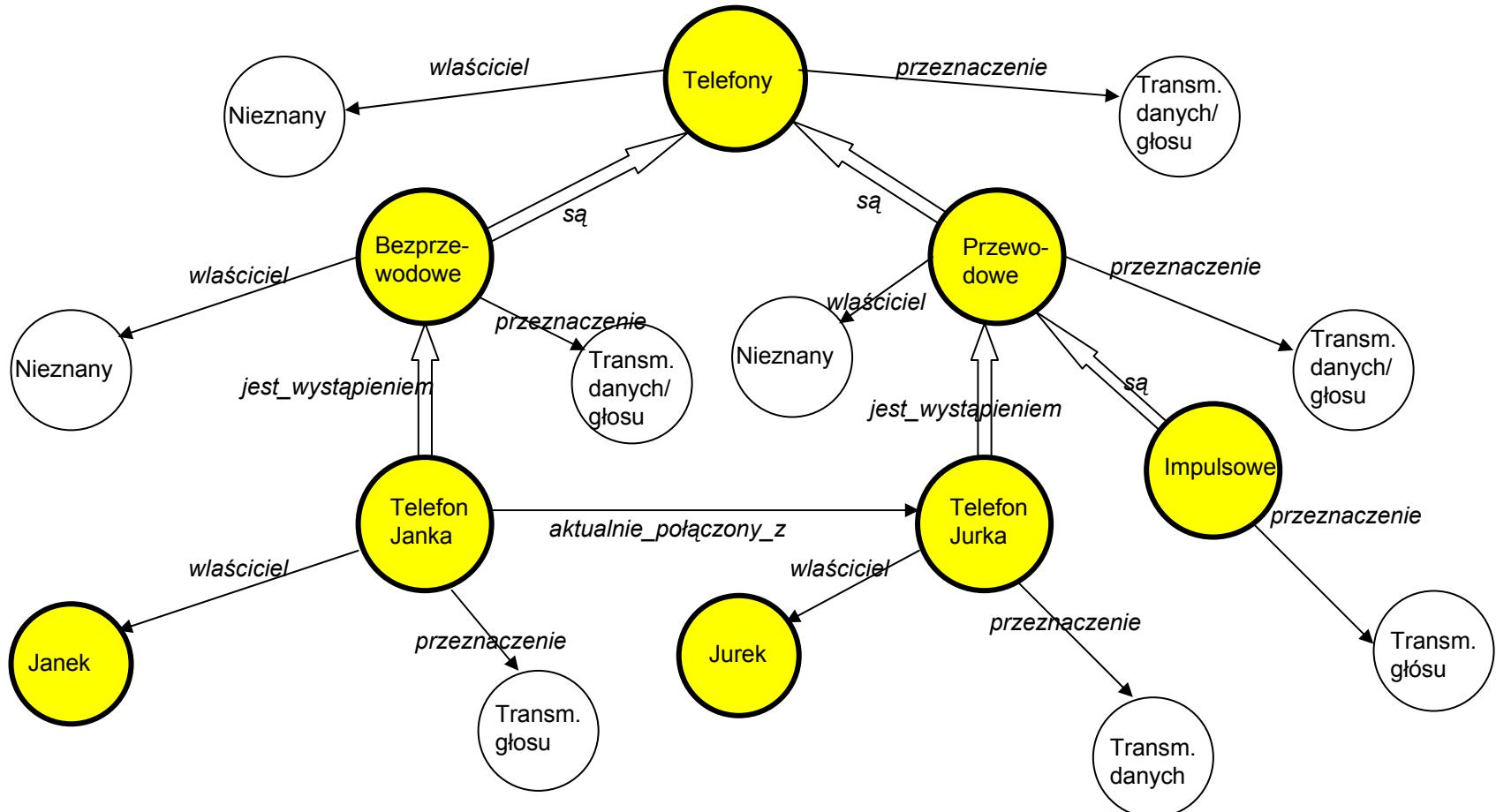
formalny język graficzny umożliwiający reprezentowanie faktów dotyczących obiektów świata rzeczywistego.

Fakty

- istnienie
- hierarchia abstrakcji pomiędzy klasami lub klasami i obiektami
- hierarchia właściwości pomiędzy obiektami lub obiektami i wartościami

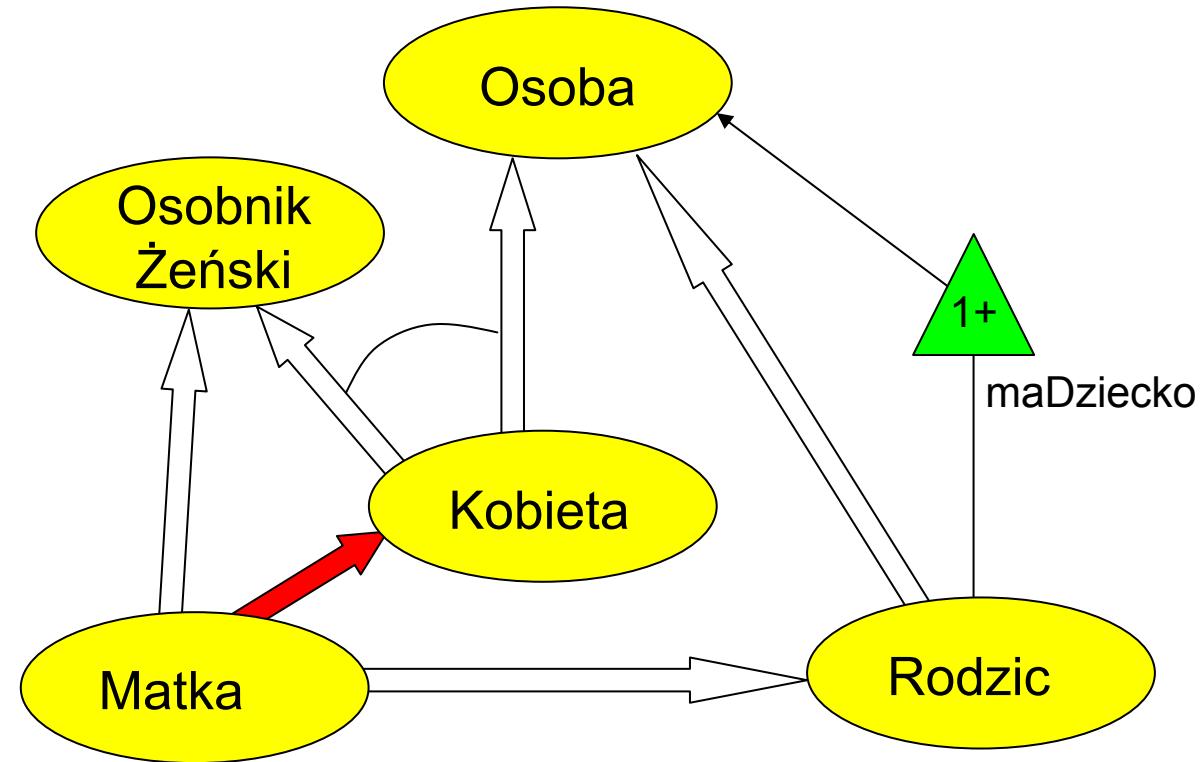
Sieć semantyczna jest graficznym zapisem asercji logicznych.

Sieci semantyczne - przykład



W zależności od przyjętego modelu, podklasa dziedziczy wszystkie atrybuty lub istnieją atrybuty niedziedziczone (dopuszczalne są „wyjątki”).

Sieć semantyczna

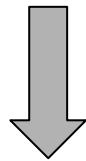


Jaki jest związek pomiędzy Matką a Kobietą?



Wady ramek i sieci semantycznych

- Brak jednej, ścisłej teorii, a co za tym idzie – brak jednoznacznej interpretacji.
- Trudności we wnioskowaniu (pojęcie metaklasy prowadzi do nierozstrzygalności wnioskowania).
- Bardzo duża ogólność („opis wszystkiego”).



Trudności w reprezentowaniu
i przetwarzaniu przez komputery



Spis treści

1. Wprowadzenie do ontologii
2. Wczesne ontologiczne metody reprezentacji wiedzy
- 3. Semantyczny Internet (Semantic Web)**
4. Resource Description Framework (RDF)
5. Web Ontology Language (OWL)
6. Logika opisowa (DL) jako fundament ontologicznych metod reprezentacji wiedzy
7. Inżynieria ontologii
8. Źródła





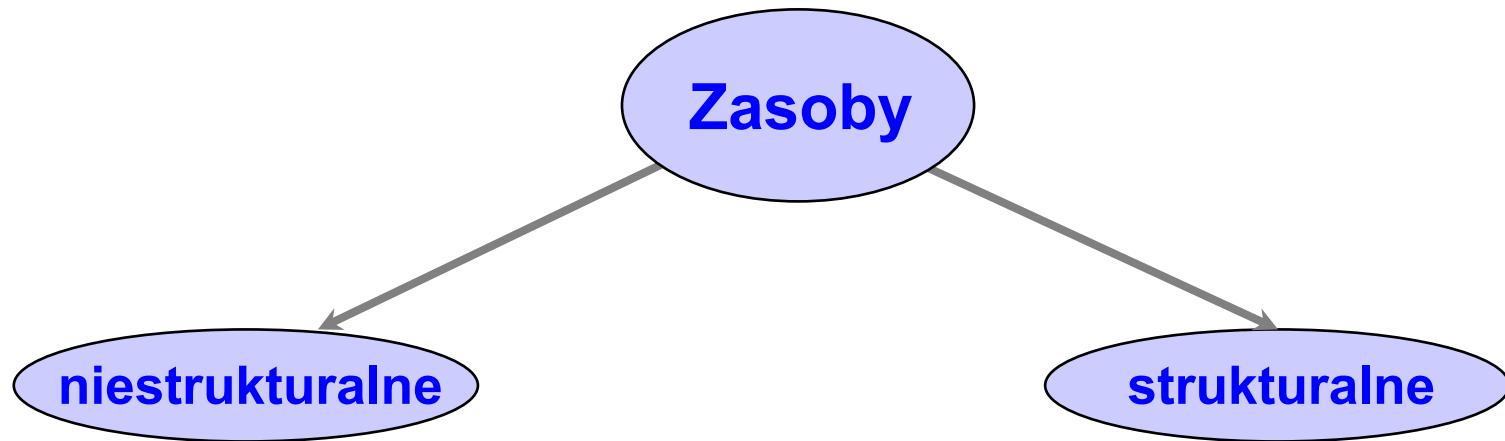
Nastała era Internetu...

- Olbrzymie zasoby informacji
 - różna struktura
 - różna jakość
- Łatwość dostępu do informacji
(wystarczy mieć przeglądarkę...)
- Łatwość wprowadzania własnych danych
(wystarczy mieć serwer WWW...)

Zalety czy wady ?



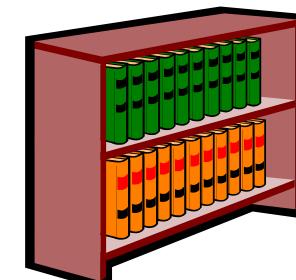
Klasyfikacja zasobów danych



nie istnieje odpowiadający im
schemat danych



istnieje ściśle zdefiniowany
schemat danych



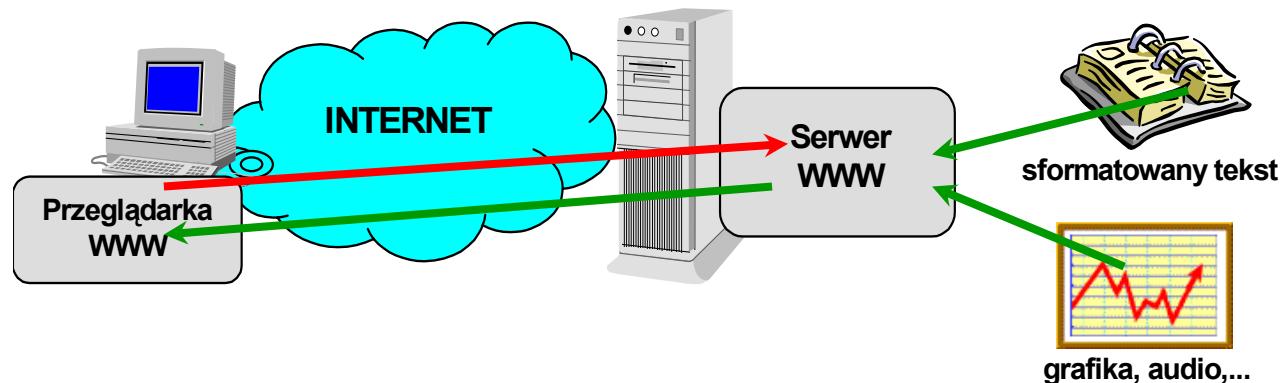
Zasoby niestrukturalne

Statyczne strony (pliki) HTML zawierające

- tekst
- grafikę
- dane audio  , wideo  ...
- odsyłacze do innych stron/plików

Identyfikowane adresem URL (*Uniform Resource Identifier*):

protokół // komputer.domena / plik#miejsce





Zasoby strukturalne

Dane zgromadzone w bazach danych dostępnych poprzez Internet

Schemat danych + Dane

Dostęp poprzez adres URL rozszerzony o parametry zapytania:

protokół // komputer.domena / procedura ? p1=w1&p2=w2 ...



Dynamiczna strona HTML



Zasoby strukturalne i niestrukturalne

Przykład:

<http://quote.yahoo.com/q?s=intc&d=t>





Zasoby semistrukturalne

Dane semistrukturalne:

- o nieznanym (lub znanym częściowo) schemacie danych
- o strukturze nieregularnej
 - pola opcjonalne
 - alternatywne kombinacje pól
 - pola o nieokreślonej liczności
- niekompletne
- często zawierające błędy i niespójności

Język do opisu i zapisu tego typu danych:

eXtensible Markup Language (XML)



Zasadnicze pytania

Jak to wszystko zintegrować?

Jak to wszystko uczynić możliwym do przetwarzania przez komputer (*machine-readable*)?



Semantic Web (Semantyczny Internet)

- **Semantic Web -**

Rozszerzenie istniejącej sieci WWW o mechanizmy semantyczne, tak aby informacje dostępne w tej sieci były dobrze zdefiniowane i umożliwiały lepszą współpracę komputerom i ludziom.

T.Berners-Lee, J.Hendler, O.Lassila. "The Semantic Web", Scientific American, 2001.

- **Elementy Semantic Web:**

- Ontologie
- Bazy wiedzy
- Środowiska agentowe



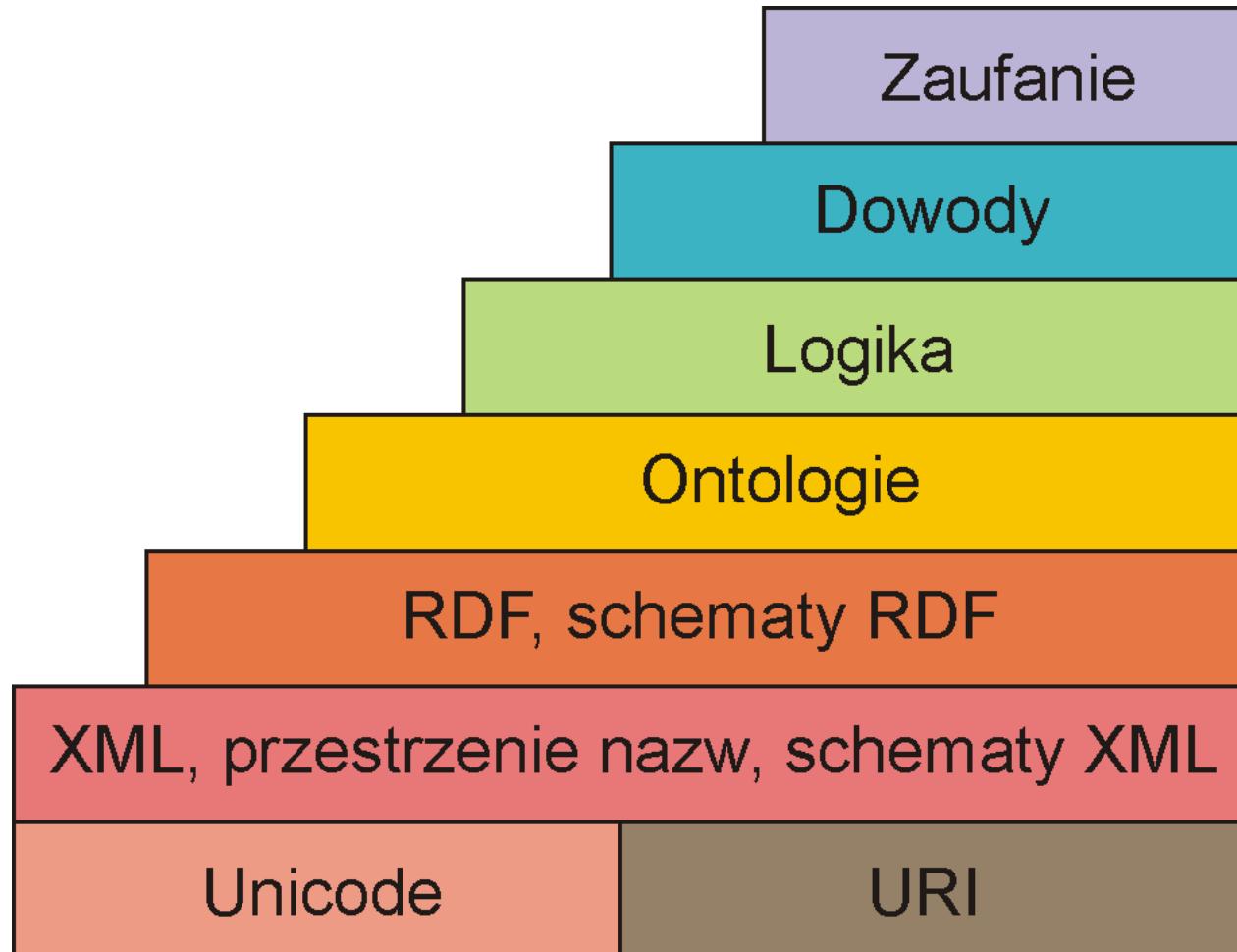
World Wide Web Consortium (W3C)

- Organizacja zajmująca się ustanawianiem standardów tworzenia i przesyłu stron WWW.
 - Założona w 1994 roku przez Tima Berners-Lee, wynalazcę WWW, autora pierwszej przeglądarki internetowej i serwera WWW.
 - W3C jest zrzeszeniem ponad 360 organizacji, firm, agencji rządowych i uczelni z całego świata.
 - Publikowane przez W3C rekomendacje nie mają mocy prawnej, ale często stają się standardami *de facto*.

www.w3.org



„Torcik” Semantic Web



www.semanticweb.org



Składniki Semantic Web

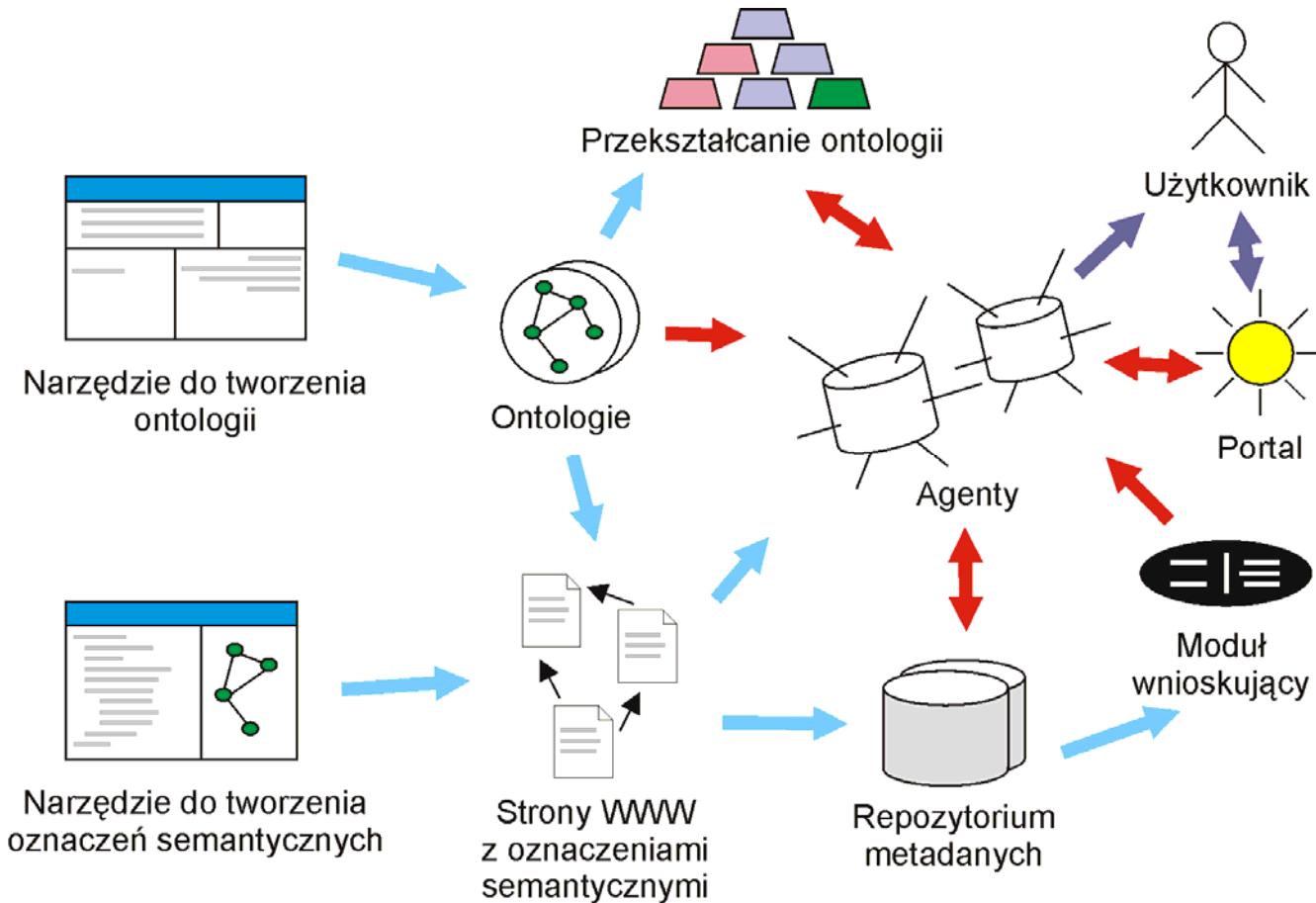
- **XML**
 - baza syntaktyczna
- **RDF**
 - RDF podstawowy model danych dla faktów
 - RDF Schema: reguły tworzenia prostych ontologii
- **Słownictwo ontologiczne**
 - bardziej ekspresywne języki
 - standardy W3C (OWL)



Składniki Semantic Web

- **Logika**
 - ontologie (encyklopedie), słowniki, ...
 - reguły proceduralne
- **Dowody**
 - wnioskowanie, wymiana informacji, walidacja
- **Zaufanie**
 - Podpisy cyfrowe
 - Instytucje rankingowe, rekomendacje, ...
 - Bezpieczeństwo

„Łańcuch pokarmowy” Semantic Web



Społeczeństwo wiedzy (?)



Języki reprezentacji wiedzy

1. Resource Description Framework (RDF)
jako język do zapisu informacji
o zasobach Internetu.

2. Web Ontology Language (OWL) jako język
do tworzenia ontologii Semantic Web.



Spis treści...

1. Wprowadzenie do ontologii
2. Wczesne ontologiczne metody reprezentacji wiedzy
3. Semantyczny Internet (Semantic Web)
- 4. Resource Description Framework (RDF)**
5. Web Ontology Language (OWL)
6. Logika opisowa (DL) jako fundament ontologicznych metod reprezentacji wiedzy
7. Inżynieria wiedzy
8. Źródła





Iinicjatywa Semantic Web i RDF

- Iinicjatywa Semantic Web zmierza ku utworzeniu jednolitych standardów opisu zasobów sieciowych, tak aby ich semantyka była jasna dla różnych narzędzi przetwarzających informacje,
- W ramach tej inicjatywy powstał szereg standardów, z których podstawowym jest RDF (ang. *Resource Description Framework*).
 - <http://www.w3.org/RDF/>
 - <http://www.w3.org/TR/rdf-primer/>



Model wiedzy RDF

Resource Description Framework - model opisu zasobów.

Zasób - wszystko, co można zidentyfikować poprzez Uniform Resource Identifier (URI).

Dokument RDF - opis zasobów i związków pomiędzy nimi.

Schemat RDF - klasy zasobów, typy związków, rodzaje ograniczeń.

Model RDF: zbiór trójek (sieć semantyczna, *Semantic Web*)





Model wiedzy RDF

Przykład:

Jan | jest_ojcem | Anny
Jan | mówi_że | (Jerzy | jest_ojcem | Anny)

Każdy element trójki jest zasobem identyfikowanym przez URI.

W celu uniknięcia konfliktów nazw wprowadza się pojęcie przestrzeni nazw (*namespace*).

Istnieją przestrzenie nazw standardowych dla RDF: rdfs, rdf, definiujące podstawowe klasy RDF:

- **rdfs:Resource** - klasa zawierająca wszystkie zasoby
- **rdfs:Class** - klasa zawierająca wszystkie klasy i ich instancje
- **rdf:Property** - klasa zawierająca wszystkie właściwości
- **rdf:Restriction** - klasa zawierająca wszystkie ograniczenia

Inne przestrzenie nazw:

- **Dublin Core (dublincore.org)**



RDF - przykład



```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
          xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">  
  <contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">  
    <contact:fullName>Eric Miller</contact:fullName>  
    <contact:mailbox rdf:resource="mailto:em@w3.org"/>  
    <contact:personalTitle>Dr.</contact:personalTitle>  
  </contact:Person>  
</rdf:RDF>
```



RDFS - Resource Description Framework Schema

- **Wnioskowanie:**

Jeżeli A jest podklassą klasy B i a jest obiektem typu A,
to a jest obiektem typu B

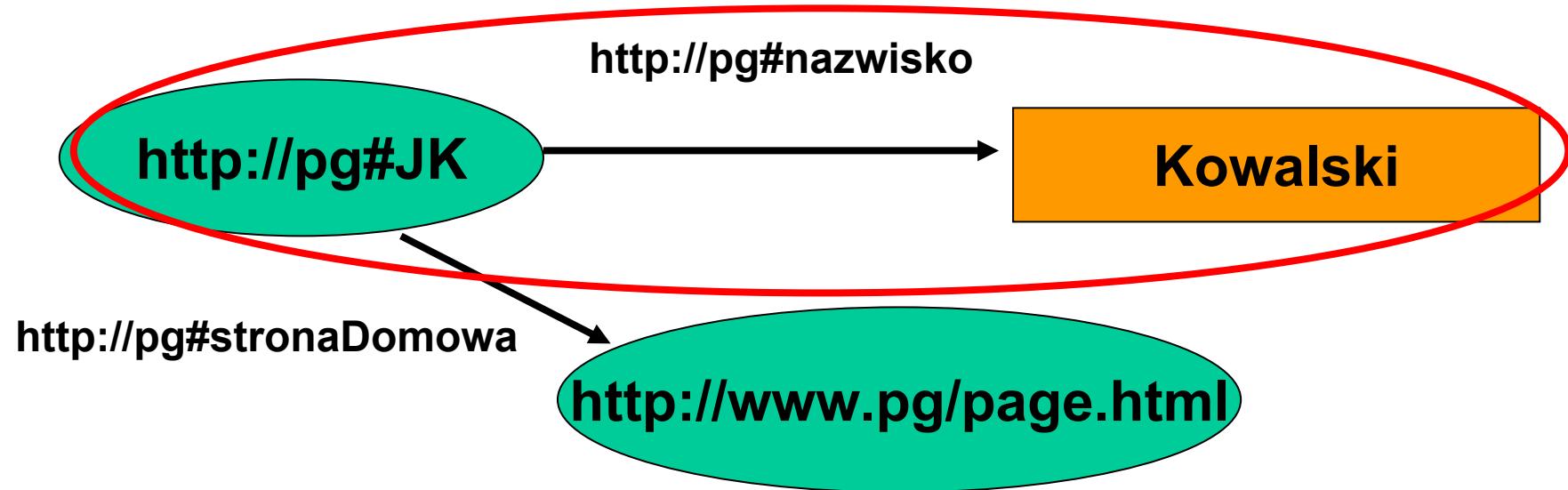
$$\left. \begin{array}{l} (\text{A}, \text{rdfs:subClassOf}, \text{B}) \\ (\text{a}, \text{rdf:type}, \text{A}) \end{array} \right\} \Rightarrow (\text{a}, \text{rdf:type B})$$

- **Narzędzie:**

Jena 2 Toolkit (<http://jena.sourceforge.net>)



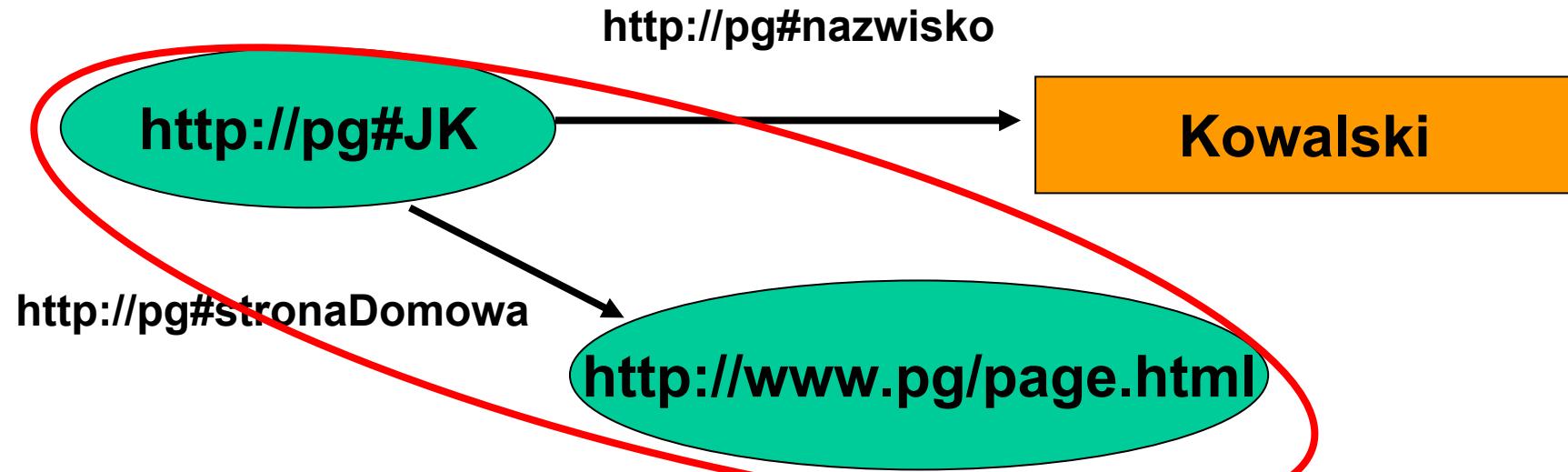
RDF – trójkąt (1)



```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:pg="http://pg#">
    <rdf:Resource rdf:about="http://pg#JK">
        <pg:nazwisko>Kowalski</pg:nazwisko>
        <pg:stronaDomowa>
            <rdf:Resource rdf:about="http://www.pg/page.html"/>
        </pg:stronaDomowa>
    </rdf:Resource>
</rdf:RDF>
```



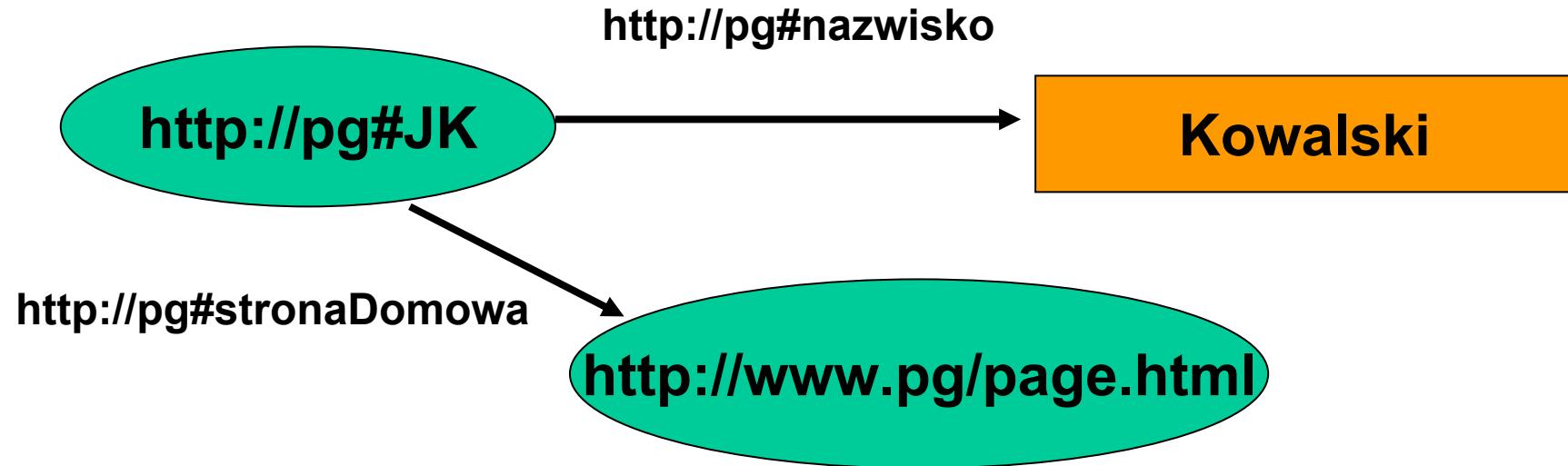
RDF – trójkąt (2)



```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:pg="http://pg#">
    <rdf:Resource rdf:about="http://pg#JK">
        <pg:nazwisko>Kowalski</pg:nazwisko>
        <pg:stronaDomowa>
            <rdf:Resource rdf:about="http://www.pg/page.html"/>
        </pg:stronaDomowa>
    </rdf:Resource>
</rdf:RDF>
```



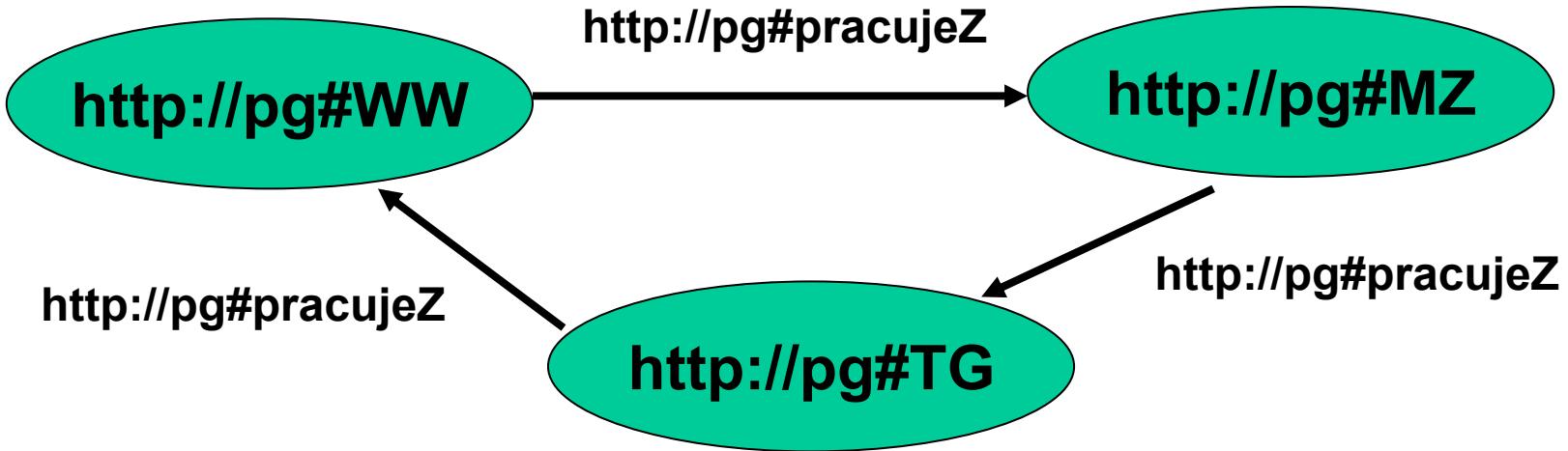
RDF – skrócony zapis



```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:pg="http://pg#"
    >
    <rdf:Resource rdf:about="http://pg#JK"
        pg:nazwisko="Kowalski">
        <pg:stronaDomowa rdf:resource="http://www.pg/page.html"/>
    </rdf:Resource>
</rdf:RDF>
```



RDF – grafy



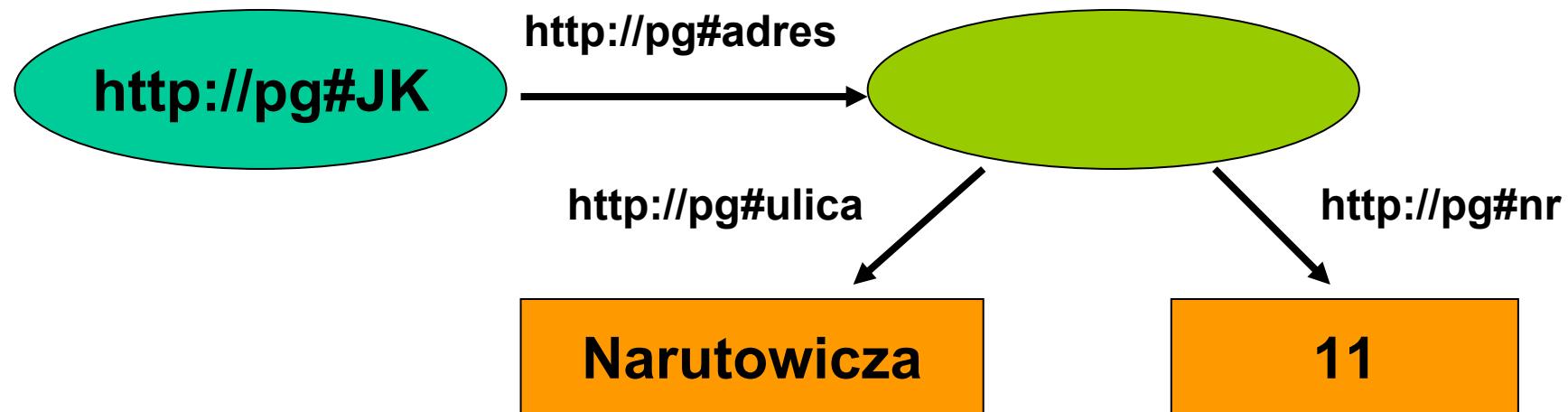
...

```
<rdf:Resource rdf:about="http://pg#WW">
    <pg:pracujeZ rdf:resource="http://pg#MZ"/>
</rdf:Resource>
<rdf:Resource rdf:about="http://pg#MZ">
    <pg:pracujeZ rdf:resource="http://pg#TG"/>
</rdf:Resource>
<rdf:Resource rdf:about="http://pg#TG">
    <pg:pracujeZ rdf:resource="http://pg#WW"/>
</rdf:Resource>
```

...



RDF – węzły anonimowe

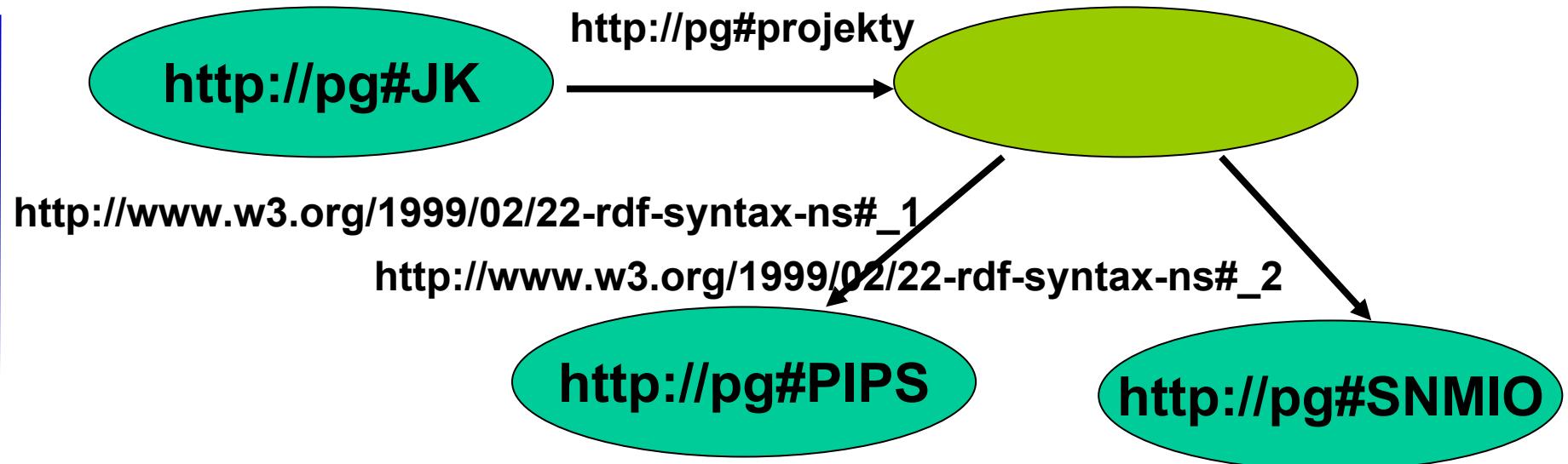


...

```
<rdf:Resource rdf:about="http://pg#JK">
  <pg:adres>
    <rdf:Resource>
      <pg:ulica>Narutowicza</pg:ulica>
      <pg:nr>11</pg:nr>
    <rdf:Resource>
  </pg:adres>
</rdf:Resource>
...
```



RDF – kontenery



...

```
<rdf:Resource rdf:about="http://pg#JK">
  <pg:projekty>
    <rdf:Bag>
      <rdf:li rdf:resource="http://pg#PIPS"/>
      <rdf:li rdf:resource="http://pg#SNMIO"/>
    </rdf:Bag>
  </pg:projekty>
</rdf:Resource>
...
```



RDF Schema

- Standardem towarzyszącym RDF jest *RDF Schema*, za pomocą którego można definiować własne *klasy węzłów* oraz *właściwości* (predykaty), tworząc tzw. aplikacje RDF.
- RDF Schema jest aplikacją RDF.
- W RDF Schema pomiędzy klasami oraz właściwościami można ustanawiać zależności umożliwiające **wnioskowanie**.



Przypisywanie typów

`http://www.w3.org/1999/02/22-rdf-syntax-ns#type`

`http://pg#JK`

`http://pg#Informatyk`

```
...
<rdf:Resource rdf:about="http://pg#JK">
  <rdf:type rdf:resource="http://pg#Informatyk"/>
</rdf:Resource>
...
...
<pg:Informatyk rdf:about="http://pg#JK"/>
...
```



Definiowanie typów

`http://www.w3.org/1999/02/22-rdf-syntax-ns#type`

`http://pg#Pracownik`

`...rdf-syntax-ns:Class`

`http://www.w3.org/1999/02/22-rdf-syntax-ns#subClassOf`

`http://pg#Informatyk`

`...rdf-syntax-ns:Class`

`http://www.w3.org/1999/02/22-rdf-syntax-ns#type`

...

```
<rdf:Class rdf:about="http://pg#Pracownik"/>
```

```
<rdf:Class rdf:about="http://pg#Informatyk">
```

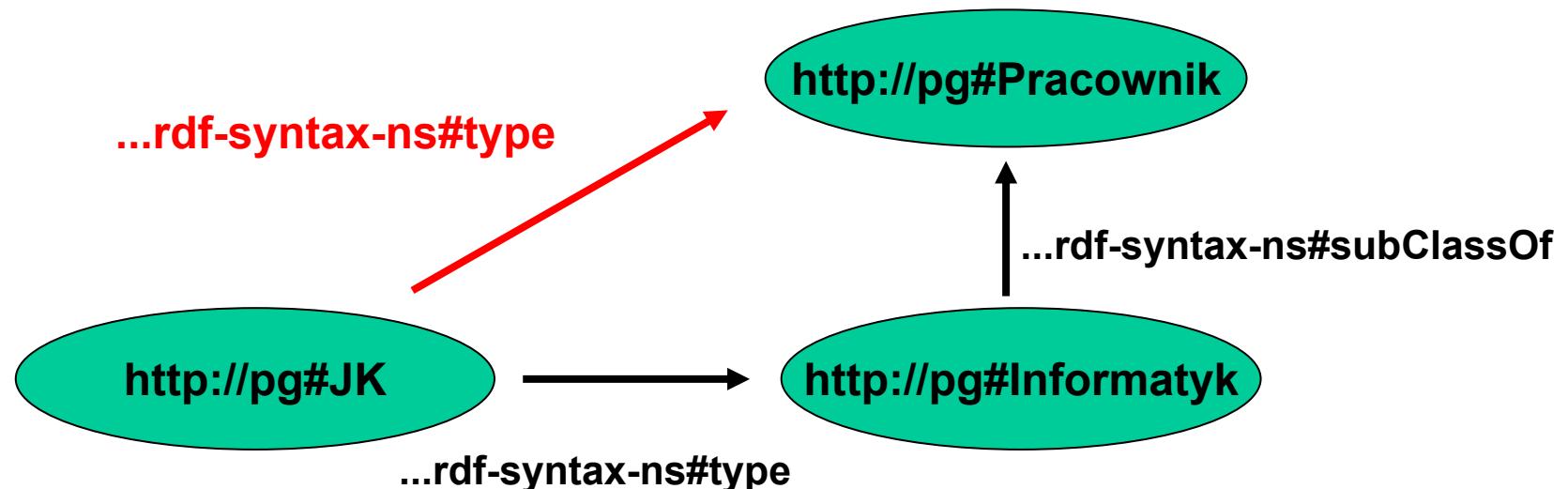
```
  <rdf:subClassOf rdf:resource="http://pg#Pracownik"/>
```

```
  </rdf:Class>
```

...

Wnioskowanie

- Wnioskowanie w RDF polega na odkrywaniu nowych prawdziwych trójkę na podstawie tych, które już znamy.



```
...
<pg:Informatyk rdf:about="http://pg#JK"/>
...
```



Wnioskowanie i OWL

- Dzięki wnioskowaniu możliwe jest łączenie różnych opisów w spójne porcje informacji na temat interesujących nas zasobów sieciowych.
- Znacznie większe możliwości w zakresie opisu zależności między klasami (a co za tym idzie w zakresie wnioskowania) daje OWL (ang. *Web Ontology Language*)
 - <http://www.w3.org/2004/OWL/>



Spis treści

1. Wprowadzenie do ontologii
2. Wczesne ontologiczne metody reprezentacji wiedzy
3. Semantyczny Internet (Semantic Web)
4. Resource Description Framework (RDF)
- 5. Web Ontology Language (OWL)**
6. Logika opisowa (DL) jako fundament ontologicznych metod reprezentacji wiedzy
7. Źródła





Web Ontology Language (OWL)

- Zaprojektowany do opisywania i wymiany wiedzy
- Standard rekomendowany przez W3C
- Rozwijany w ramach inicjatywy Semantic Web
- Oparty na logice opisowej (DL)





Trzy dialekty OWL

- W3C zdefiniowało OWL jako trzy różne podjęzyki:
 - OWL Full
 - OWL DL
 - OWL Lite
- Każdy podjęzyk opracowany został w celu spełnienia różnych aspektów wymagań





OWL Full

- Używa wszystkich podstawowych konstrukcji języków OWL
- Pozwala na kombinacje tych konstrukcji z konstrukcjami RDF i RDF Schema
- OWL Full jest w pełni kompatybilny w górę z RDF, zarówno syntaktycznie, jak i semantycznie
- OWL Full jest tak potężny, że jest nierozstrzygalny
 - Nie istnieje pełne (lub wydajne) wsparcie dla wnioskowania



OWL DL

- OWL DL (Description Logic) jest podjęzykiem OWL Full, który ogranicza użycie konstruktorów z OWL i RDF
 - Niedozwolone jest stosowanie konstruktorów OWL wobec siebie
 - Odpowiada logice opisowej
- OWL DL zapewnia efektywne wnioskowanie
- Utara pełnej kompatybilności z RDF:
 - Nie każdy poprawny dokument RDF jest poprawnym dokumentem OWL DL.
 - Każdy poprawny dokument OWL DL jest poprawnym dokumentem RDF.



OWL Lite

- Ograniczenie OWL DL do podzbioru konstruktorów tego języka
 - Np. OWL Lite nie dopuszcza klas wyliczeniowych, rozłączności czy ograniczeń liczebnościowych.
- Zaletą tego języka jest łatwość
 - zrozumienia – dla użytkowników
 - implementacji – dla twórców narzędzi
- Wadą jest ograniczona ekspresyjność



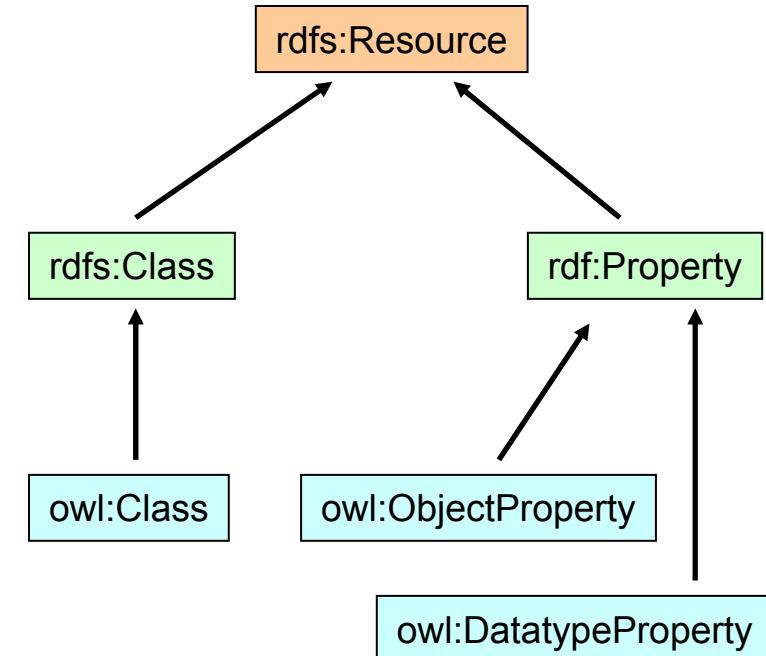
Kompatybilność między dialektami

- Każda poprawna ontologia OWL Lite jest poprawną ontologią OWL DL
- Każda poprawna ontologia OWL DL jest poprawną ontologią OWL Full
- Każdy poprawny wniosek w OWL Lite jest poprawnym wnioskiem w OWL DL
- Każdy poprawny wniosek w OWL DL jest poprawnym wnioskiem w OWL Full



Kompatybilność OWL z RDF Schema

- Wszystkie dialekty OWL używają składni RDF
- Osobniki są deklarowane tak jak w RDF - używając opisów RDF
- Konstruktory OWL są uszczegółowieniem ich odpowiedników w RDF





Konstrukcje OWL (1)

- Właściwości

- przechodniość



```
<owl:TransitiveProperty rdf:id="subRegionOf">
  <rdfs:domain rdf:resource="#Region"/>
  <rdfs:range  rdf:resource="#Region"/>
</owl:TransitiveProperty>
```

- symetryczność



```
<owl:SymmetricProperty rdf:id="friendOf">
  <rdfs:domain rdf:resource="#Human"/>
  <rdfs:range  rdf:resource="#Human"/>
</owl:SymmetricProperty>
```

Konstrukcje OWL (2)

- Właściwości

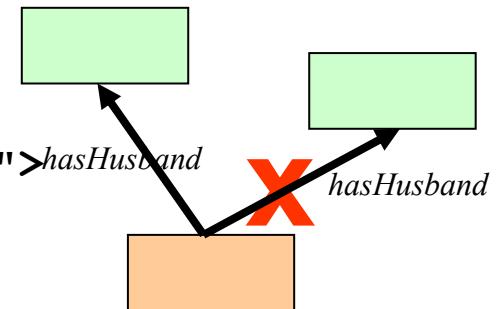
- odwrotność

```
<owl:ObjectProperty rdf:ID="hasChild">  
  <owl:inverseOf rdf:resource="#hasParent"/>  
</owl:ObjectProperty>
```



- funkcjonalność

```
<owl:FunctionalProperty rdf:ID="hasHusband">  
  <rdfs:domain rdf:resource="#Woman" />  
  <rdfs:range rdf:resource="#Man" />  
</owl:FunctionalProperty>
```



- odwrotna funkcjonalność

```
<owl:InverseFunctionalProperty rdf:ID="MotherOf">  
  <rdfs:domain rdf:resource="#Woman"/>  
  <rdfs:range rdf:resource="#Human"/>  
</owl:InverseFunctionalProperty>
```



Konstrukcje OWL (3)

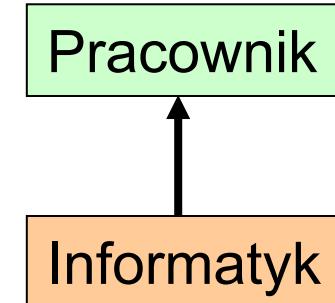
- **Zakresy właściwości**
 - obiekty (ObjectProperty)
 - wartości (DataProperty)
- **Dziedziny właściwości**
 - tylko obiekty
- **Cecha obiektu (feature):**
 - właściwość funkcjonalna o zakresie typu ObjectProperty
- **Atrybut obiektu (attribute):**
 - właściwość funkcjonalna o zakresie typu DataProperty.
- **Inne konstrukcje**
 - Dziedziny konkretne (liczby, łańcuchy, ...)
 - Przestrzenie nazewnicze



Konstrukcje OWL (4)

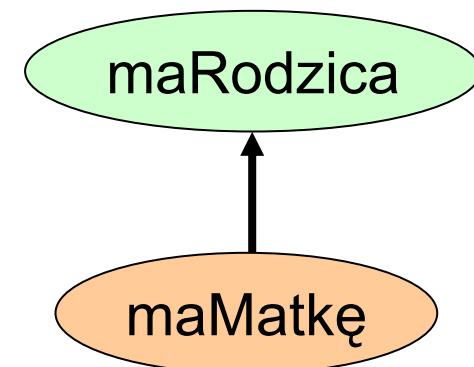
- Dziedziczenie (zawieranie się) klas

```
<owl:Class rdf:ID="Informatyk">
  <rdfs:subClassOf
    rdf:resource="#Pracownik" />
</owl:Class>
```



- Dziedziczenie (zawieranie się) właściwości

```
<owl:ObjectProperty rdf:ID="maMatkę">
  <rdfs:subPropertyOf
    rdf:resource="#maRodzica"/>
</owl:ObjectProperty>
```



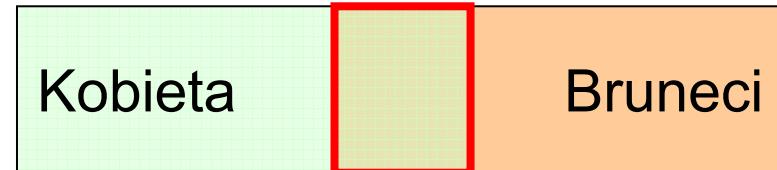


Konstrukcje OWL (5)

- Operatory zbiorowe na klasach

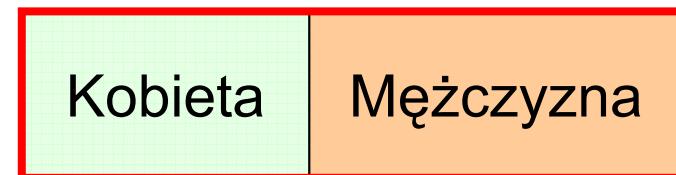
- przecięcie

```
<owl:Class rdf:ID="Brunetki">
  <owl:intersectionOf
    rdf:parseType="Collection">
    <owl:Class rdf:resource="Kobieta">
    <owl:Class rdf:resource="Bruneci">
  </owl:intersectionOf>
</owl:Class>
```



- unia

```
<owl:Class rdf:ID="Człowiek">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:resource="#Kobieta">
    <owl:Class rdf:resource="#Mężczyzna">
  </owl:unionOf>
</owl:Class>
```

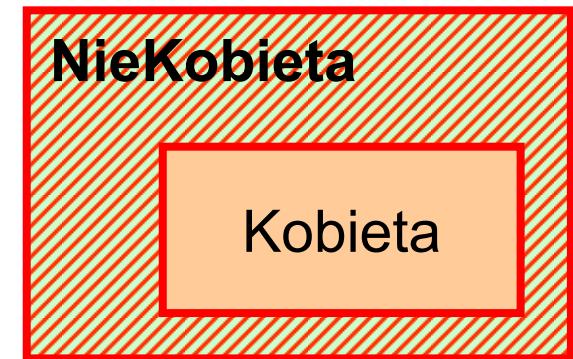




Konstrukcje OWL (6)

- Operatory zbiorowe na klasach
 - dopełnienie

```
<owl:Class rdf:ID="NieKobieta">
  <owl:complementOf>
    <owl:Class rdf:resource="Kobieta">
  </owl:complementOf>
</owl:Class>
```



- Ograniczenia liczebnościowe właściwości

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#maRodzica" />
  <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">
    2
  </owl:cardinality>
</owl:Restriction>
```

```
<owl:minCardinality />
```

```
<owl:maxCardinality />
```



Moduły i importowanie

- Obsługa importowania w OWL jest bardzo prosta:

- Pozwala jedynie na import całej ontologii – a nie jej części

...

```
<owl:Ontology>
  <owl:imports
    rdf:resource="http://www.pg/other.owl"/>
</owl:Ontology>
...
```



Przykład OWL (1)



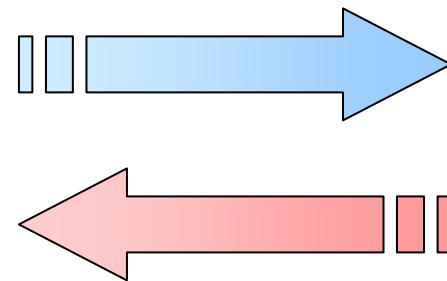
```
<owl:Class rdf:ID="Pacjent"/>
```

```
<owl:Class rdf:ID="PomiarCisnienia"/>
```

```
<owl:ObjectProperty rdf:ID="maBadanie">
  <owl:domain rdf:resource="#Pacjent"/>
  <owl:range rdf:resource="#PomiarCisnienia"/>
</owl:ObjectProperty>
```



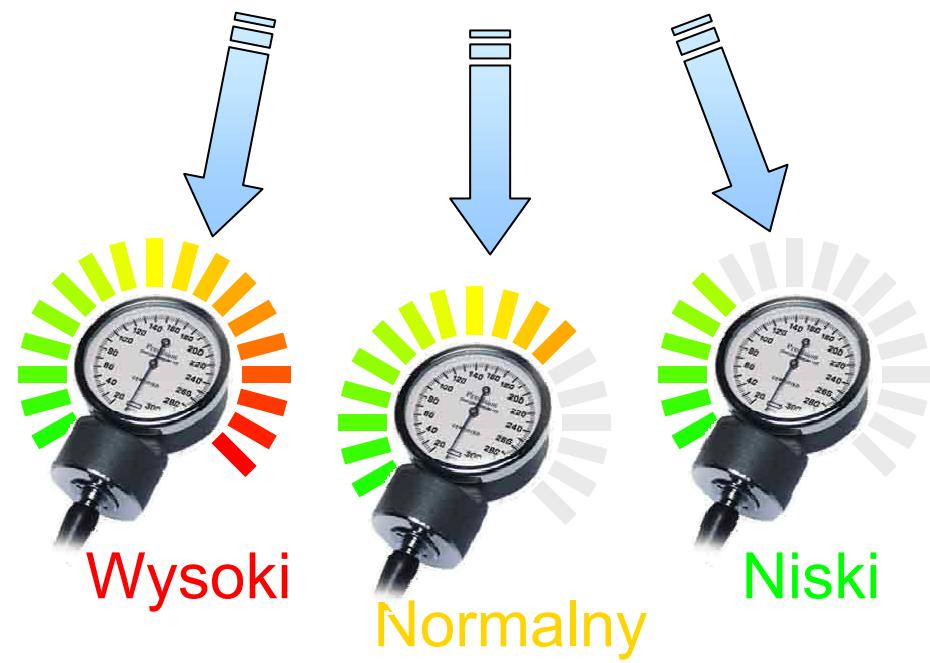
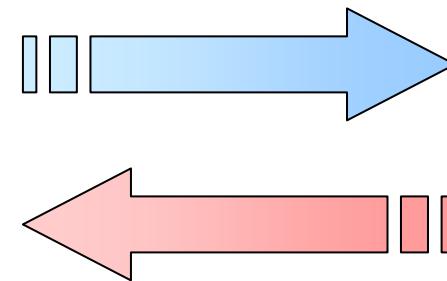
Przykład OWL (2)



```
<owl:ObjectProperty rdf:ID="dotyczyPacjenta">
  <owl:domain rdf:resource="#PomiarCisnienia"/>
  <owl:range rdf:resource="#Pacjent"/>
  <owl:inverseOf rdf:resource="#maBadanie"/>
</owl:ObjectProperty>
```



Przykład OWL (3)



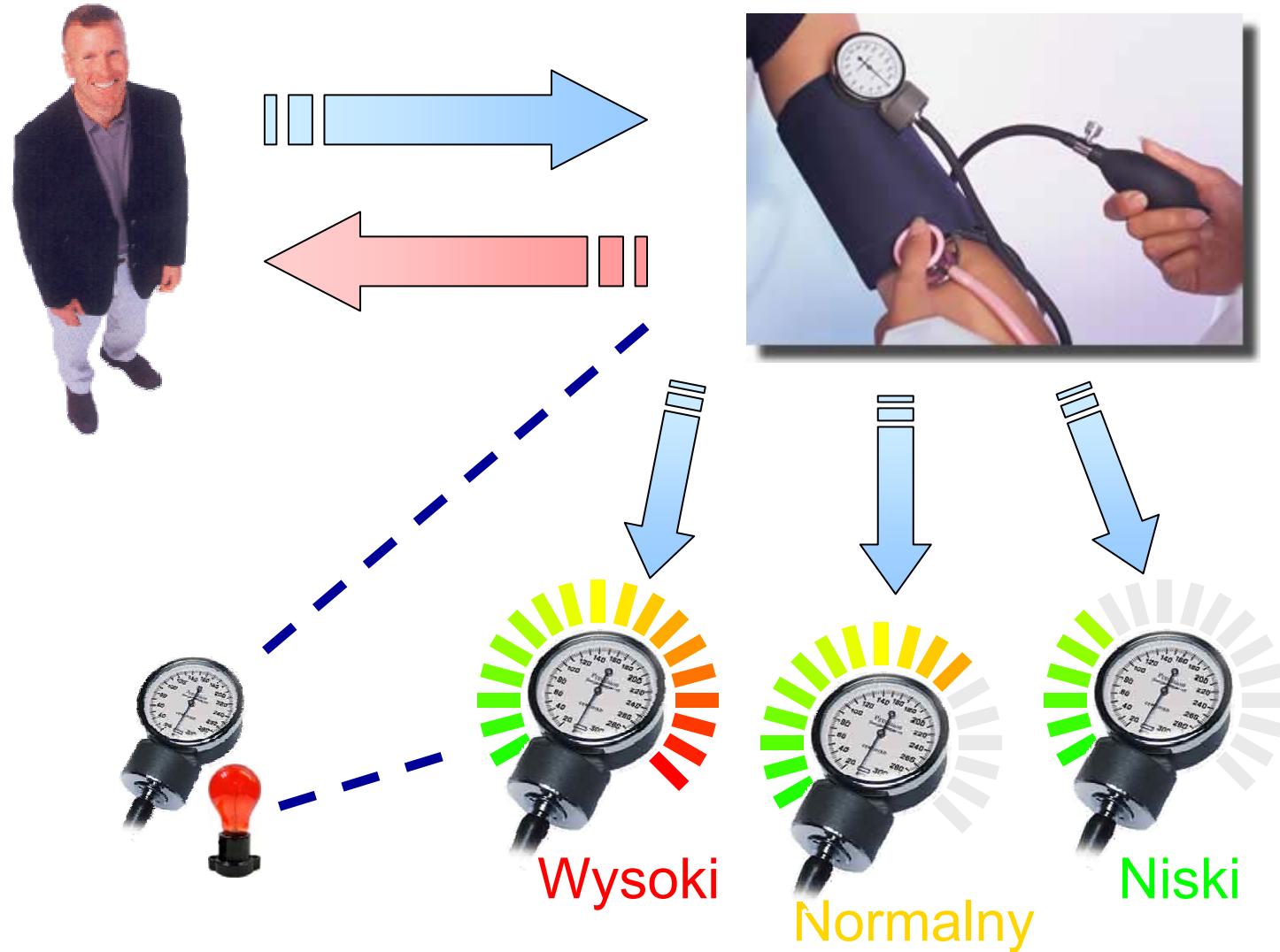


Przykład OWL (3)

```
<owl:Class rdf:ID="WynikPomiaruCisnienia">
  <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:about="#Niski"/>
    <owl:Thing rdf:about="#Normalny"/>
    <owl:Thing rdf:about="#Wysoki"/>
  </owl:oneOf>
</owl:Class>

<owl:ObjectProperty rdf:ID="maWynik">
  <owl:domain rdf:resource="#PomiarCisnienia"/>
  <owl:range rdf:resource="#WynikPomiaruCisnienia"/>
</owl:ObjectProperty>
```

Przykład OWL (4)



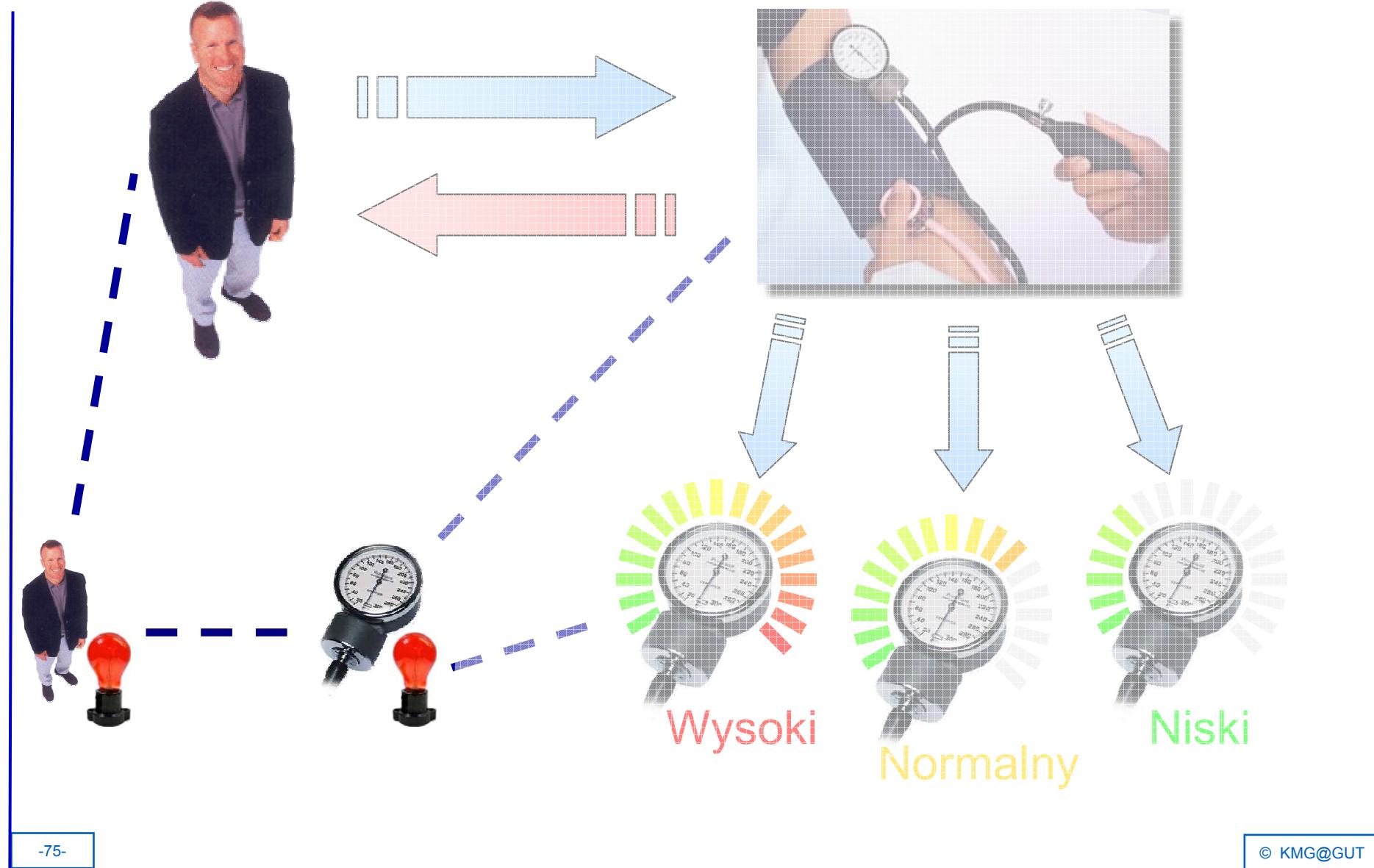


Przykład OWL (4)

```
<owl:Class rdf:id="PomiarCisnieniaWykazujacyNadcisnienie">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#PomiarCisnienia"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#maWyniki"/>
      <owl:hasValue rdf:resource="#Wysoki"/>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```



Przykład OWL (5)

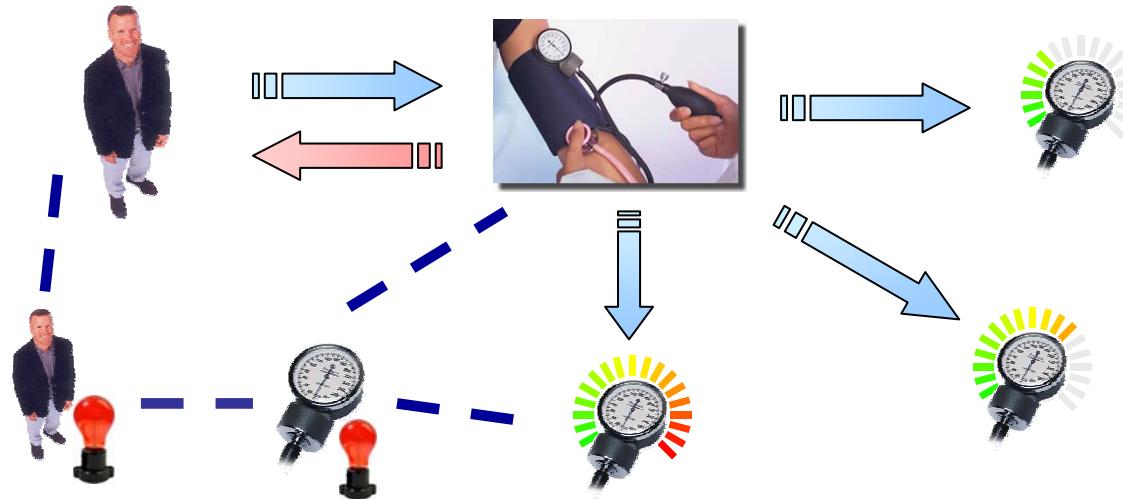




Przykład OWL (5)

```
<owl:Class rdf:ID="PacjentZagrozonyNadcisnieniem">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Pacjent"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#maBadanie"/>
      <owl:someValuesFrom
        rdf:resource="#PomiarCisnieniaWykazujacyNadcisnienie"/>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

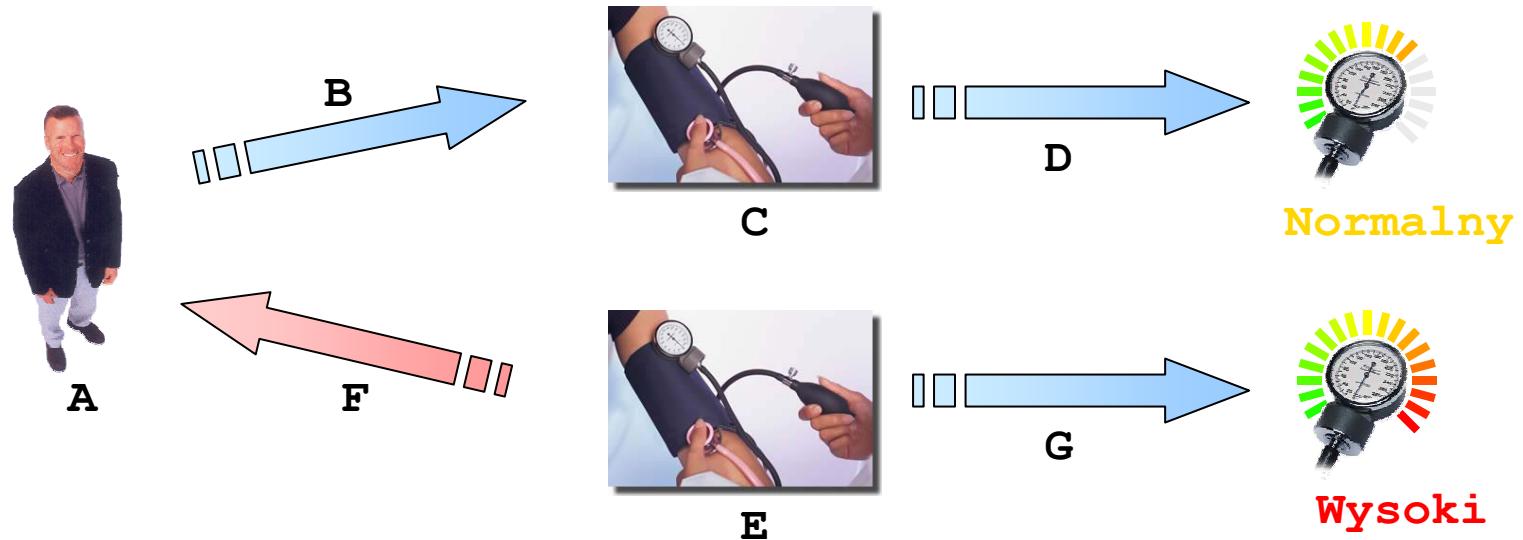
Terminologia



1. $\text{maBadanie}(X, Y) \Leftrightarrow \text{dotyczyPacjenta}(Y, X)$
2. $\text{PomiarCisnieniaWykazujacyNadcisnienie}(X) \Leftrightarrow \text{PomiarCisnienia}(X) \wedge \text{maWynik}(X, \#Wysoki)$
3. $\text{PacjentZagrozonyNadcisnieniem}(X) \Leftrightarrow \text{Pacjent}(X) \wedge \exists Y(\text{maBadanie}(X, Y) \wedge \text{PomiarCisnieniaWykazujacyNadcisnienie}(Y))$

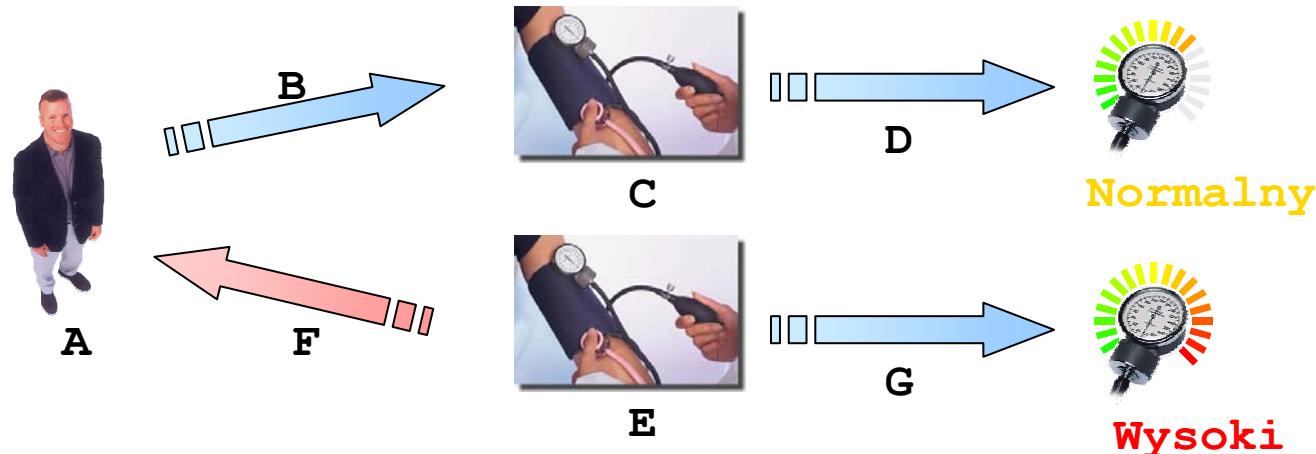
Opis Świata

- A. Pacjent(#JanKowalski)
- B. maBadanie(#JanKowalski, #BadanieJK1)
- C. PomiarCisnienia(#BadanieJK1)
- D. maWynik(#BadanieJK1, #Normalny)
- E. PomiarCisnienia(#BadanieJK2)
- F. dotyczyPacjenta(#BadanieJK2, #JanKowalski)
- G. maWynik(#BadanieJK2, #Wysoki)





Opis Świata



```
<Pacjent rdf:id="JanKowalski"> A  
  <maBadanie rdf:resource="#BadanieJK1"/> B  
</Pacjent>  
<PomiarCisnienia rdf:id="BadanieJK1"> C  
  <maWynik rdf:resource="#Normalny"/> D  
</PomiarCisnienia>  
<PomiarCisnienia rdf:id="BadanieJK2"> E  
  <dotyczyPacjenta rdf:resource="#JanKowalski"/> F  
  <maWynik rdf:resource="#Wysoki"/> G  
</PomiarCisnienia>
```



Ważne pytanie

Czy Jan Kowalski jest zagrożony nadciśnieniem?

PacjentZagrożonyNadcisnieniem (#JanKowalski) ?



Wnioskowanie (1)

I. maBadanie(#JanKowalski, #BadanieJK2)

z F i 1:

dotyczyPacjenta(#BadanieJK2, #JanKowalski) \Rightarrow
maBadanie(#JanKowalski, #BadanieJK2)



II. PomiarCisnieniaWykazujacyNadcisnienie(#BadanieJK2)

z E, G i 2:

PomiarCisnienia(#BadanieJK2) \wedge
maWynik(#BadanieJK2, #Wysoki) \Rightarrow
PomiarCisnieniaWykazujacyNadcisnienie(#BadanieJK2)





Wnioskowanie (2)

III. PacjentZagrozonyNadcisnieniem(#JanKowalski)

z I, II, A i 3:

```
Pacjent( #JanKowalski ) ∧  
hasTest( #JanKowalski, #BadanieJK2 ) ∧  
PomiarCisnieniaWykazujacyNadcisnienie( #BadanieJK2 ) ⇒  
PacjentZagrozonyNadcisnieniem( #JanKowalski )
```





Wnioskowanie (2)



Jan Kowalski jest zagrożony nadciśnieniem!



Spis treści...

Logika opisowa (Description Logic, DL) jako fundament ontologicznych metod reprezentacji wiedzy

1. Ontologia i baza wiedzy DL
2. Konstruktory DL i języki opisu
3. Wnioskowanie z terminologii i opisu świata
4. Ekspresywne logiki opisowe
5. DL a OWL
6. OWA a CWA



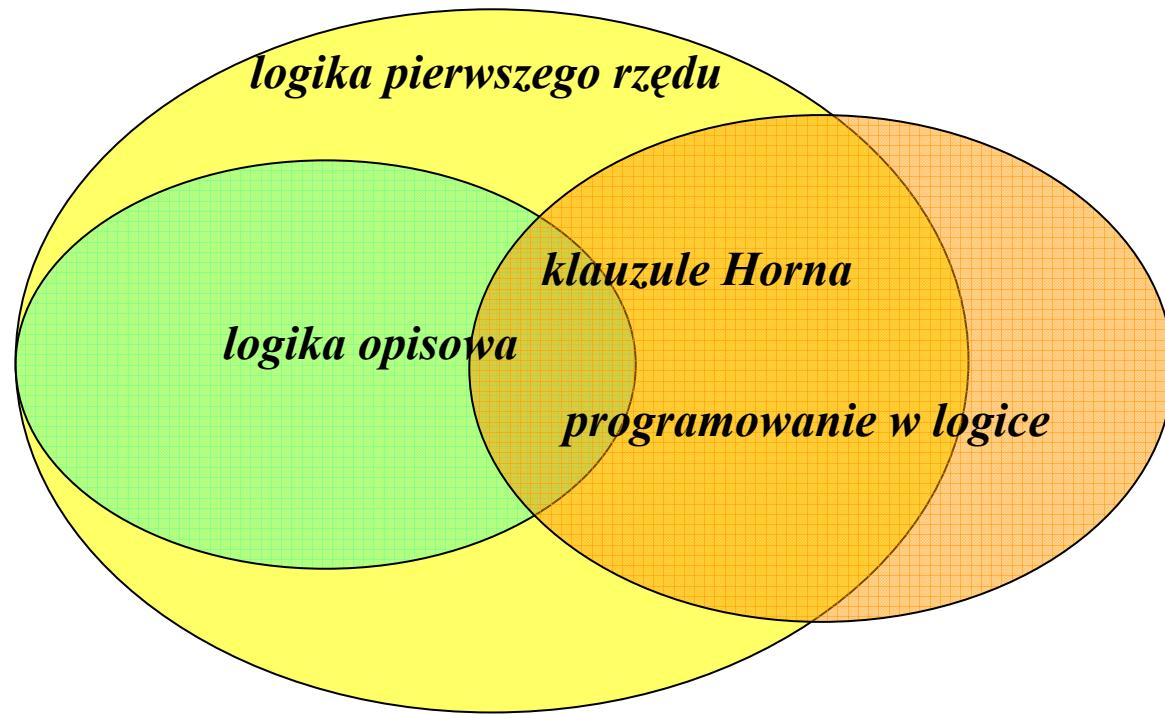


Logika opisowa (*Description Logic - DL*)

- Logika opisowa – formalizm służący do definiowania ontologii, oparty na rachunku predykatów pierwszego rzędu.
- Logika opisowa jest (rozstrzygalnym) podzbiorem logiki pierwszego rzędu (FOL):
 - brak pojęcia funkcji
 - wszystkie relacje są albo unarne, albo binarne
 - zazwyczaj domyślnie przyjmuje się założenie o unikatowości nazw (*Unique Name Assumption – UNA*)



Logika pierwszego rzędu, logika opisowa, klauzule Horna, programowanie w logice



First Order Logic (FOL)
Description Logic (DL)
Horn Logic (HL)
Logic Programming (LP)



Logika opisowa (cd.)

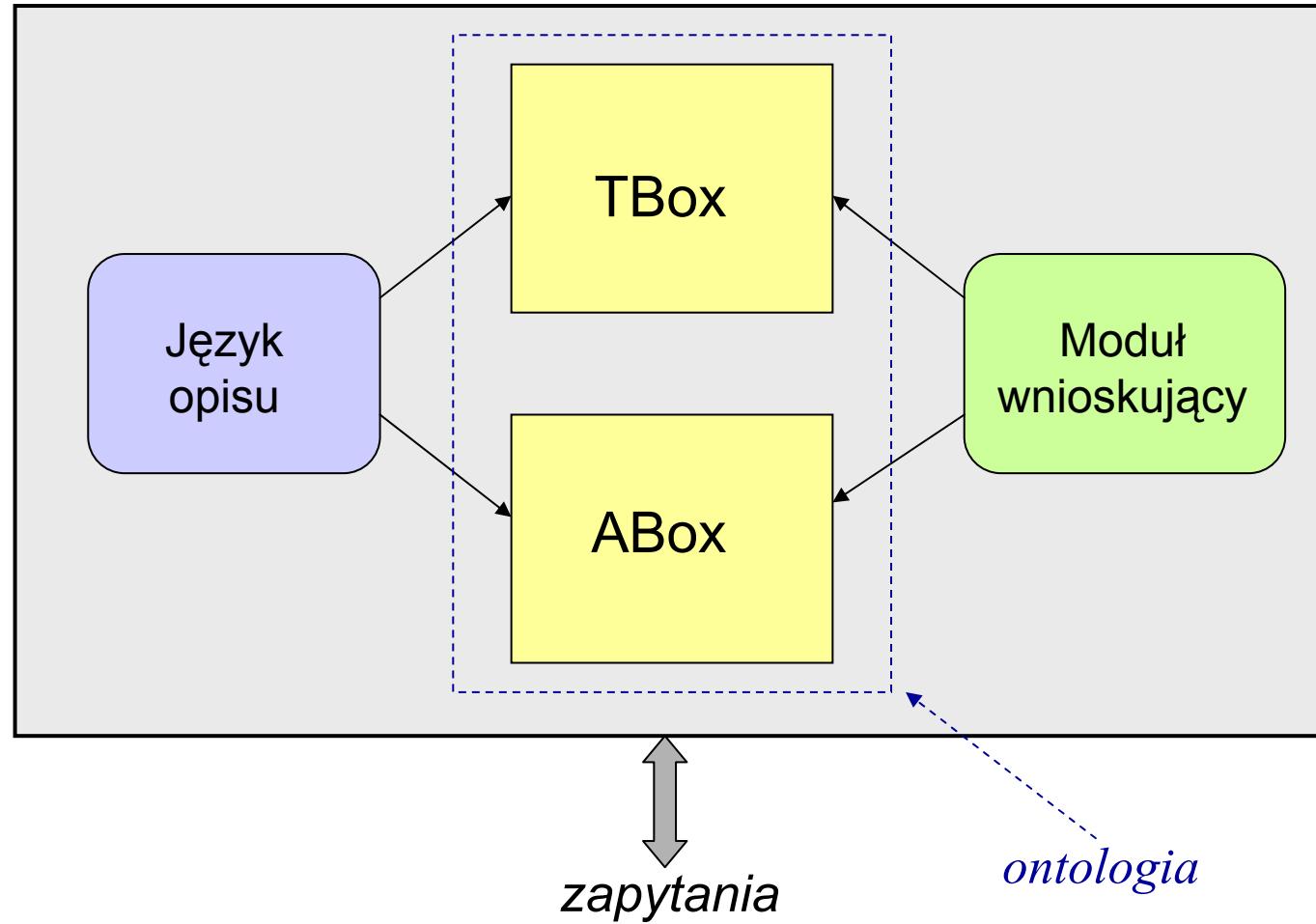
Logika opisowa obejmuje rodzinę języków do definiowania ontologii o różnym stopniu ekspresji, dających różne możliwości wnioskowania.

Założenia:

- *Istnieje uniwersum, które chcemy opisać.*
- *Elementy tego uniwersum („osobniki”, „indywidua”) są wystąpieniami konceptów.*
- *Koncepty są ze sobą powiązane.*



Baza wiedzy



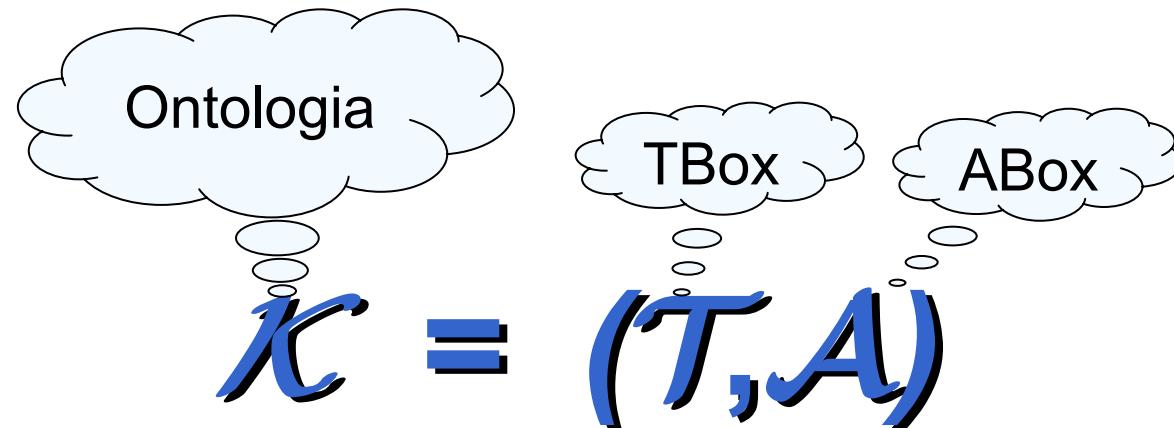


Uniwersum

- Określa naszą dziedzinę zainteresowań, pośrednio wskazując zbiór osobników; wiedzę o tych osobnikach i relacjach między nimi będziemy przetwarzać.
- Przykłady:
 - Mieszkania i lokatorzy spółdzielni mieszkaniowej
 - Ekosystem sawanny
 - Studenci, nauczyciele, przedmioty
 - ...



Ontologia - składniki



TBox (terminologia) – zbiór *koncepcji*,
zbiór binarnych relacji pomiędzy nimi (*ról*),
zbiór ograniczeń (*aksjomatów*).

ABox (opis świata) – zbiór *asercji unarnych*
(opisujących wystąpienia konceptów) i
asercji binarnych (opisujących wystąpienia ról).



Terminologia

- Wprowadza system nazw, za pomocą których możemy nazywać osobniki naszego uniwersum (**koncepty**), a także zależności pomiędzy nimi (**role**).
- Określa zależności pomiędzy terminami w postaci wyrażeń logicznych – **aksjomatów**.



Język opisu ontologii \mathcal{ALC}

Podstawowe elementy:

- koncepty atomowe
 - w tym: *koncepcja uniwersalna* \top (*Top*)
 - koncepcja pusta* \perp (*Bottom*)
- role atomowe
- konstruktory konceptów i ról

Konstruktory języka \mathcal{ALC} :

$\neg C$	- negacja konceptu
$C \sqcap D$	- przecięcie konceptów
$C \sqcup D$	- suma konceptów
$\exists R.C$	- kwantyfikacja egzystencjalna
$\forall R.C$	- kwantyfikacja ogólna (ograniczenie wartości)



Kwantyfikacje

$\exists R.C$ - oznacza zbiór takich osobników, które są powiązane przynajmniej jeden raz rolą R z osobnikiem będącym wystąpieniem konceptu C .

Przykład: $\exists \text{maDziecko.Kobieta}$

Jest to koncept oznaczający te osoby, które mają przynajmniej jedną córkę.

$\forall R.C$ - oznacza zbiór takich osobników, których wszystkie **istniejące** powiązania rolą R dotyczą osobników będących wystąpieniem konceptu C .

Przykład: $\forall \text{maDziecko.Kobieta}$

Jest to koncept oznaczający te osoby, których wszystkie dzieci są córkami.



Kwantyfikacje (2)

Uwaga:

Wystąpieniami konceptu $\forall \text{maDziecko.Kobieta}(x) \Leftrightarrow (\text{maDziecko}(x, y) \Rightarrow \text{Kobieta}(y))$ Kobieta są także takie osobniki, które nie mają żadnych dzieci. Wynika to ze znaczenia kwantyfikacji ogólnej w logice pierwszego rzędu (FOL).

!

„Wszystkie worki moich pieniędzy leżą na tym stole”



$$\forall \text{maDziecko.Kobieta}(x) \Leftrightarrow (\text{maDziecko}(x, y) \Rightarrow \text{Kobieta}(y))$$

$$\forall R.C \equiv \neg \exists R. \neg C$$



Przykłady konceptów złożonych

- Koncepty złożone pozwalają na wskazanie grup osobników niewyróżnionych poprzez nazwy atomowe:

Lokator \sqcap CzłonekSpółdzielni

Drapieżnik \sqcap $\neg \exists$ jada.Drapieżnik

Student \sqcap \forall maOcena.OcenaWysoka



Aksjomaty

- *Aksjomaty* określają zależności pomiędzy wprowadzonymi koncepcjami,
- Wyróżniamy *aksjomaty podrzędnosci* (\sqsubseteq) oraz *aksjomaty równoważności* (\equiv):

CzłonekSpółdzielni \sqsubseteq Lokator

Drapieżnik \equiv \exists zjada.Zwierzę

Student \sqsubseteq \exists Tutora.Nauczyciel

- Aksjomaty pozwalają też na określenie właściwości ról – m.in. ich *dziedzin* i *zakresów*



Dziedziny i zakresy ról

Dziedzina roli –

Koncept, którego wystąpieniami są osobniki będące pierwszym argumentem asercji (*role owners*).

Definicja aksjomatyczna dziedziny roli R :

$$\exists R. \top \sqsubseteq Dziedzina$$

Przykład:

Dziedziną roli *maMęża* jest *Kobieta*.

$$\exists maMęża. \top \sqsubseteq Kobieta$$



Dziedziny i zakresy ról

Zakres roli –

Koncept, którego wystąpieniami są osobniki będące drugim argumentem asercji (*role fillers*).

Definicja aksjomatyczna zakresu roli R :

$$\exists R. \neg \text{Zakres} \sqsubseteq \perp$$

lub równoważnie:

$$\forall R. \text{Zakres} \sqsupseteq \top$$

Przykład:

Zakresem roli *maMęża* jest *Mężczyzna*.

$$\exists \text{maMęża}. \neg \text{Mężczyzna} \equiv \perp$$



Opis świata

- Pozwala na „ujawnianie” fragmentów uniwersum i opisywanie ich za pomocą konceptów i ról zdefiniowanych w terminologii,
- Składa się z *asercji unarnych* (stwierdzających przynależność osobnika do kontekstu) i *asercji binarnych* (stwierdzających wystąpienie roli pomiędzy parą osobników):

Zwierzę(Tygrys)

Roślinożerca(Antylopa)

zjada(Tygrys, Antylopa)



Problemy wnioskowania

Podstawowe problemy wnioskowania z terminologii:

- **subsumcji** (*subsumption*):

Czy zbiór wystąpień konceptu C jest zawsze podzbiorem wystąpień konceptu D ? $(C \sqsubseteq D)$

- **spełnialność** (*satisfiability*):

Czy koncept C może mieć wystąpienia? $(C \equiv \perp)$

- **równoważność** (*equivalence*):

Czy zbiory wystąpień konceptów C i D są zawsze równe? $(C \equiv D)$

- **rozłączność** (*disjointness*):

Czy zbiory wystąpień konceptów C i D są zawsze rozłączne? $(C \sqcap D \equiv \perp)$



Problemy wnioskowania (2)

Problemy wnioskowania z terminologii nie są od siebie niezależne.
Na przykład: pozostałe trzy problemy można sprowadzić do
problemu subsumcji:

- **spełnialność (satisfiability):**

Czy $C \sqsubseteq \perp$?

- **równoważność (equivalence):**

Czy $C \sqsubseteq D$ i jednocześnie $D \sqsubseteq C$?

- **rozłączność (disjointness):**

Czy $C \sqcap D \sqsubseteq \perp$?



Problemy wnioskowania (3)

Podstawowe problemy wnioskowania z terminologii i opisu świata:

- **określenie zbioru wystąpień konceptu** (*retrieval*):
Jakie osobniki należą do konceptu C?
instances (C)
- **sprawdzenie przynależności** (*instance check*):
Czy dany osobnik należy do konceptu C?
instance (x, C)
- **sprawdzenie spójności** (*consistency check*):
Czy baza wiedzy nie zawiera sprzeczności?
(Czy istnieje niepusty **model** bazy wiedzy?)
consistent (KB)



Przykład bazy wiedzy

TBox

Mężczyzna ⊑ Osoba

Kobieta ⊑ Osoba

Kobieta □ Mężczyzna ≡ ⊥

Rodzic ≡ Osoba □ ∃maDziecko.Osoba

Ojciec ≡ Mężczyzna □ Rodzic

Matka ≡ Kobieta □ Rodzic



ABox

Kobieta (Anna)

Kobieta (Joanna)

Mężczyzna (Karol)

maDziecko (Anna, Joanna)

maDziecko (Anna, Karol)

Niejawną rozłączność:

Matka □ Ojciec ≡ ⊥

Matka □ Mężczyzna ≡ ⊥

...

Niejawne zawieranie się:

Matka ⊑ Kobieta

Matka ⊑ Osoba

...



Przykład bazy wiedzy

TBox

Mężczyzna ⊑ Osoba

Kobieta ⊑ Osoba

Kobieta □ Mężczyzna ≡ ⊥

Rodzic ≡ Osoba □ ∃maDziecko.Osoba

Ojciec ≡ Mężczyzna □ Rodzic

Matka ≡ Kobieta □ Rodzic



ABox

Kobieta (Anna)

Kobieta (Joanna)

Mężczyzna (Karol)

maDziecko (Anna, Joanna)

maDziecko (Anna, Karol)

- Opiekun ≡ Matka □ Ojciec

Opiekun to koncept niespełnialny

- Kobieta (Maria)
Mężczyzna (Maria)

Maria i Jan to osobniki bez konceptu
(baza wiedzy stała się sprzeczna)

- Rodzic (Jan)
¬Rodzic (Jan)



Wnioskowanie z ontologii

TBox

Mężczyzna ⊑ Osoba

Kobieta ⊑ Osoba

Kobieta □ Mężczyzna ≡ ⊥

Rodzic ≡ Osoba □ ∃maDziecko.Osoba

Ojciec ≡ Mężczyzna □ Rodzic

Matka ≡ Kobieta □ Rodzic



ABox

Kobieta (Anna)

Kobieta (Joanna)

Mężczyzna (Karol)

maDziecko (Anna, Joanna)

maDziecko (Anna, Karol)

Pytanie 1: instances (Matka)

Odpowiedź: Anna

Pytanie 2: instances (\neg Matka)

Odpowiedź: Karol



Wnioskowanie z ontologii

TBox

Mężczyzna ⊑ Osoba

Kobieta ⊑ Osoba

Kobieta □ Mężczyzna ≡ ⊥

Rodzic ≡ Osoba □ ∃maDziecko.Osoba

Ojciec ≡ Mężczyzna □ Rodzic

Matka ≡ Kobieta □ Rodzic



ABox

Kobieta (Anna)

Kobieta (Joanna)

Mężczyzna (Karol)

maDziecko (Anna, Joanna)

maDziecko (Anna, Karol)

Pytanie 3: *instance (Anna, Matka)*

Odpowiedź: *true*

Pytanie 4: *instance (Joanna, Matka)*

Odpowiedź: *don't know (maybe...)*
(nie można dowieść ani prawdziwości, ani fałszywości)



Wnioskowanie z ontologii

TBox

Mężczyzna ⊑ Osoba

Kobieta ⊑ Osoba

Kobieta □ Mężczyzna ≡ ⊥

Rodzic ≡ Osoba □ ∃maDziecko.Osoba

Ojciec ≡ Mężczyzna □ Rodzic

Matka ≡ Kobieta □ Rodzic

maSyna ⊑ maDziecko

∃maSyna.¬Mężczyzna ≡ ⊥



ABox

Kobieta (Anna)

Kobieta (Joanna)

Mężczyzna (Karol)

maDziecko (Anna, Joanna)

maDziecko (Anna, Karol)

maSyna (Karol, Jan)

Pytanie 5: types (Jan)

Odpowiedź: Mężczyzna, Osoba



Ograniczenia liczbowościowe (*cardinality constraints*)

$\geq n R$

Koncept oznaczający zbiór osobników, które są powiązane rolą R z co najmniej n osobnikami.

$\leq n R$

Koncept oznaczający zbiór osobników, które są powiązane rolą R z co najwyżej n osobnikami.

Przykład:

OsobaSredniodzietna \equiv Osoba $\sqcap \geq 2$ maDziecko $\sqcap \leq 4$ maDziecko

oznacza osoby, które mają 2, 3 lub 4 dzieci (dowolnej płci)



Kwalifikowane ograniczenia liczebnościowe (qualified cardinality constraints)

$\geq n R.C$

Koncept oznaczający zbiór osobników, które są powiązane rolą R z co najmniej n osobnikami będącymi wystąpieniami konceptu C .

$\leq n R.C$

Koncept oznaczający zbiór osobników, które są powiązane rolą R z co najwyżej n osobnikami będącymi wystąpieniami konceptu C .

Przykład:

$\text{OsobaZCórkami} \equiv \text{Osoba} \sqcap \geq 2 \text{ maDziecko.Kobieta}$

oznacza osoby, które mają córki.



Dziedziny i zakresy ról

Dziedzina roli –

Koncept, którego wystąpieniami są osobniki będące pierwszym argumentem asercji (*role owners*).

Przykład:

Dziedziną roli *maMęża* jest *Kobieta*.

OWL

znacznik `<owl: domain .../>`

DL

Definicja aksjomatyczna dziedziny roli R :

$$\exists R. \top \sqsubseteq Dziedzina$$



Dziedziny i zakresy ról

Zakres roli –

Koncept, którego wystąpieniami są osobniki będące drugim argumentem asercji (*role fillers*).

Przykład:

Zakresem roli *maMęża* jest *Mężczyzna*.

OWL

znacznik `<owl: range .../>`

DL

Definicja aksjomatyczna zakresu roli R :

$$\exists R. \neg \text{Zakres} \equiv \perp$$

lub:

$$\exists R^-. T \sqsubseteq \text{Zakres} \quad (R^- \text{ to rola odwrotna do } R)$$



Przykład bazy wiedzy

Terminologia (TBox):

Kobieta \sqsubseteq T

Mężczyzna \sqsubseteq T

Kobieta \equiv Mężczyzna

Opis świata (ABox):

Kobieta (Susan)

Mężczyzna (Peter)

maPrzyjaciela (John, Susan)

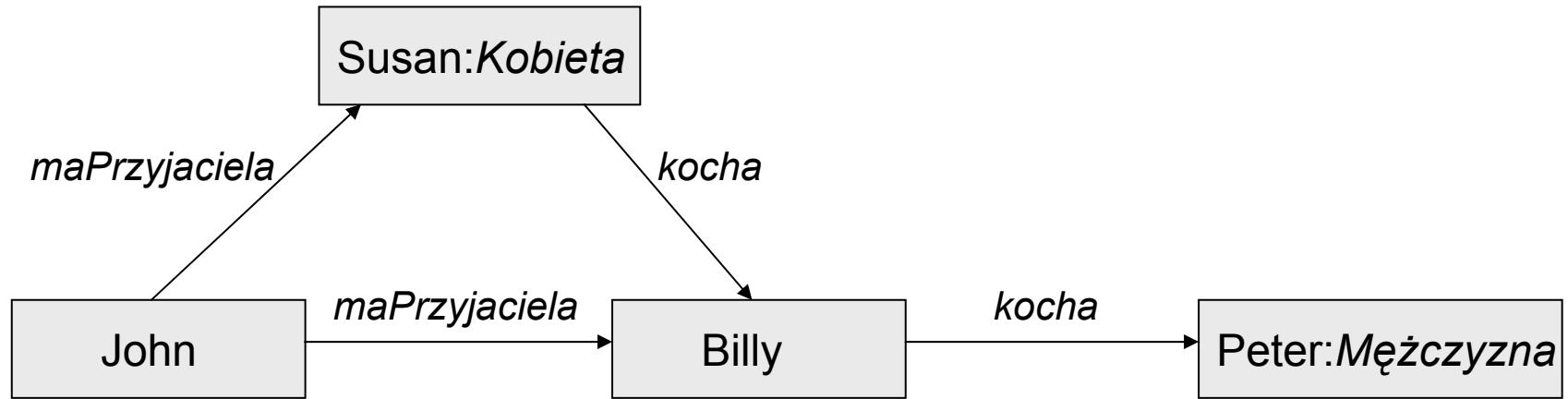
maPrzyjaciela (John, Billy)

kocha (Susan, Billy)

kocha (Billy, Peter)



Przykład bazy wiedzy

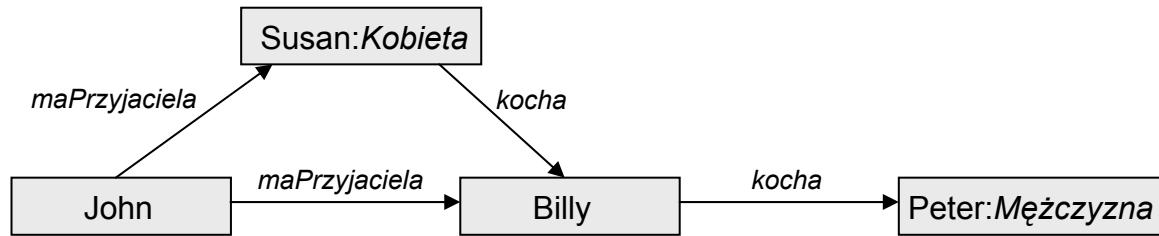


Problem:

Czy John ma przyjaciółkę, która kocha mężczyznę?



Świat zamknięty (baza danych)



Pytanie:

Czy John ma przyjaciółkę, która kocha mężczyznę?

Kobiety

imię
Susan

Mężczyźni

imię
Peter

Przyjaciele

kto	kogo
John	Susan
John	Billy

Kochankowie

kto	kogo
Susan	Billy
Billy	Peter



Świat zamknięty (baza danych)

Kobiety

imię
Susan

Mężczyźni

imię
Peter

Przyjaciele

kto	kogo
John	Susan
John	Billy

Kochankowie

kto	kogo
Susan	Billy
Billy	Peter

Czy John ma przyjaciółkę, która kocha mężczyznę?

SQL:

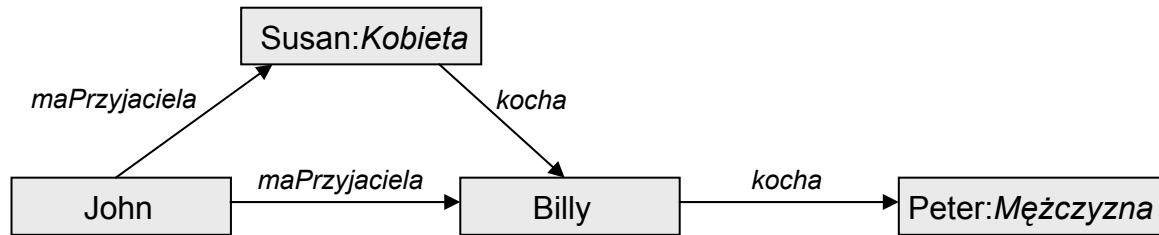
```
SELECT p.kogo
  FROM Kobiety k, Mężczyźni m, Przyjaciele p, Kochankowie c
 WHERE p.kogo = k.imię AND k.imię = c.kto AND c.kogo = m.imię
       AND p.kto = „John”
```

Wynik jest **pusty**, czyli:

NIE!



Świat otwarty (baza wiedzy)



Czy John ma przyjaciółkę, która kocha mężczyznę?

DL: Czy prawdziwa jest następująca asercja:

$(\exists \text{maPrzyjaciela}.(\text{Kobieta} \sqcap (\exists \text{kocha}. \text{Mężczyzna}))) (\text{John})$?

Wnioskowanie:

Billy jest albo mężczyzną, albo kobietą.

Jeśli Billy jest mężczyzną: **TAK!** (tą przyjaciółką jest Susan)

Jeśli Billy jest kobietą: **TAK!** (tą przyjaciółką jest Billy)



Świat otwarty a zamknięty

Która z tych odpowiedzi jest prawdziwa?
I co to znaczy „prawdziwa”?

Zauważmy, że gdybyśmy pytali o imiona przyjaciółek Johna, które kochają jakiegoś mężczyznę (*), w obu przypadkach odpowiedzią byłby zbiór pusty, bo baza wiedzy – podobnie jak baza danych - nie zna takiej osoby (w istocie, poprzednie zapytanie SQL było pytaniem właśnie o to). W odróżnieniu jednak od bazy danych, baza wiedzy wie, że taka osoba na pewno istnieje, choć jej nie zna.

Baza danych nie jest w stanie odpowiadać na pytania o istnienie czegoś – potrafi tylko odszukiwać zapisane w niej dane.

(*) *rolefillers(John, maPrzyjaciela(Kobieta \sqcap (\exists kocha.Mężczyzna)))*



OWA a CWA – przykład 2.

- Pacjent ma anginę, jeśli ma wysoką gorączkę i kaszle.
- Mary jest pacjentem.

Pytanie: Czy Mary ma anginę?

Odpowiedź: Nie – Mary nie ma wysokiej gorączki ani nie kaszle.

CWA

Odpowiedź: Nie wiem – przecież nic nie wiadomo o temperaturze ani o kaszlu Mary.

OWA



OWA – przykład 3.

Jane jest kobietą.

Helen jest kobietą.

Jane jest matką Helen.

Pytanie 1: Czy Helen ma dzieci?

Pytanie 2: Czy Jane ma wyłącznie córki?

Odpowiedź 1: Nie – Helen nie ma dzieci.

Odpowiedź 2: Tak – jedynym dzieckiem Jane jest Helen,
która jest kobietą, czyli córką Jane.

DB

Odpowiedź 1: Nie wiem – nic nie wiadomo o dzieciach
Helen.

Odpowiedź 2: Nie wiem – przecież Jane może mieć
więcej dzieci.

KB



Domykanie świata

Czasami jednak chcemy zapytać bazę wiedzy o znane dzieci (lub inne *znane* fakty).

Do tego celu służy *operator epistemologiczny K*.

Pytanie 1: Czy Helen ma dzieci?

$\exists \text{hasChild. } \top(\text{Helen})?$

Odpowiedź 1 (OWA): Nie wiem.

Pytanie 1a: Czy są znane jakieś dzieci Helen?

$\exists \mathbf{K} \text{hasChild. } \top(\text{Helen})?$

Odpowiedź 1a (CWA): Nie.

Komentarz:

Pytanie 1a jest pytaniem o to, czy w ABox istnieje choć jedna asercja postaci $\text{hasChild}(\text{Helen}, x)$, gdzie x jest dowolnym osobnikiem.



Domykanie świata (2)

Pytanie 2: Czy Jane ma wyłącznie córki?

$\forall \text{hasChild}.\text{Woman}(\text{Jane})?$

Odpowiedź 2 (OWA): Nie wiem.

Pytanie 2a: Czy wszystkie znane dzieci Jane to córki?

$\forall \text{KhasChild}.\text{Woman}(\text{Jane})?$

Odpowiedź 2a (CWA): Tak.

Komentarz:

Pytanie 2a jest pytaniem o to, czy wszystkie asercje postaci $\text{hasChild}(\text{Jane}, x)$, które są w ABox, są takie, że x jest wystąpieniem konceptu Woman .

Uwaga:

Odpowiedź na pytanie 2a brzmiałaby „Tak” także wtedy, gdyby w ABox nie było żadnej asercji postaci $\text{hasChild}(\text{Jane}, x)$.



OWA – dalsze przykłady

Ontologia:

(język naturalny)

Mary jest kobietą.
Jane jest kobietą.
Kate jest kobietą.
Mary jest matką Jane.
Mary jest matką Kate.
Jane jest bezdzietna.
Pentium4 jest procesorem.

(logika opisowa)

Woman \sqsubseteq Person
Man \sqsubseteq Person
Person \sqcap Processor $\equiv \perp$
 $\exists \text{hasChild. } \top \sqsubseteq$ Person
Woman(Mary)
Woman(Jane)
Woman(Kate)
hasChild(Mary, Jane)
hasChild(Mary, Kate)
 $\neg \exists \text{hasChild. } \top (Jane)$
Processor(Pentium4)



OWA – dalsze przykłady

Ontologia:

Woman ⊑ Person

Man ⊑ Person

Person □ Processor ≡ ⊥

∃hasChild.⊤ ⊑ Person

Woman(Mary)

Woman(Jane)

Woman(Kate)

hasChild(Mary, Jane)

hasChild(Mary, Kate)

¬∃hasChild.⊤(Jane)

Processor(Pentium4)

Pytanie:

Podaj wszystkich osobników, których wszystkie dzieci są kobietami.

instances(∀hasChild.Woman)

Odpowiedź:

{Jane, Pentium4}

Wyjaśnienie:

Wynika to ze znaczenia kwantyfikatora ogólnego (zarówno w FOL, jak i w DL):

$\forall \text{hasChild}. \text{Woman}(x) \Leftrightarrow (\text{hasChild}(x, y) \Rightarrow \text{Woman}(y))$



OWA – dalsze przykłady

Ontologia:

Woman ⊑ Person

Man ⊑ Person

Person □ Processor ≡ ⊥

∃hasChild.⊤ ⊑ Person

Woman(Mary)

Woman(Jane)

Woman(Kate)

hasChild(Mary, Jane)

hasChild(Mary, Kate)

¬∃hasChild.⊤(Jane)

Processor(Pentium4)

Pytanie:

Podaj wszystkich osobników, których wszystkie dzieci są mężczyznami.

instances(∀hasChild.Man)

Odpowiedź:

{Jane, Pentium4}

Wyjaśnienie:

Wynika to ze znaczenia kwantyfikatora ogólnego (zarówno w FOL, jak i w DL):

$\forall \text{hasChild}. \text{Man}(x) \Leftrightarrow (\text{hasChild}(x, y) \Rightarrow \text{Man}(y))$



To może inaczej?...

Ontologia:

Woman ⊑ Person

Man ⊑ Person

Person ⊓ Processor ≡ ⊥

$\exists \text{hasChild.} \top \sqsubseteq \text{Person}$

Woman(Mary)

Woman(Jane)

Woman(Kate)

hasChild(Mary, Jane)

hasChild(Mary, Kate)

$\neg \exists \text{hasChild.} \top (\text{Jane})$

Processor(Pentium4)

Pytanie:

Podaj wszystkich ludzi, którzy mają przynajmniej jedną córkę i których wszystkie dzieci są córkami.

$\text{instances}(\text{Person} \sqcap \exists \text{hasChild.} \text{Woman} \sqcap \forall \text{hasChild.} \text{Woman})$

Odpowiedź:

PUSTE!!!

Wyjaśnienie:

Przecież nie wiadomo, czy Jane i Kate są jedynymi dziećmi Mary.



Rozwiązańie: operator K

Ontologia:

Woman ⊑ Person

Man ⊑ Person

Person □ Processor ≡ ⊥

∃hasChild.⊤ ⊑ Person

Woman(Mary)

Woman(Jane)

Woman(Kate)

hasChild(Mary, Jane)

hasChild(Mary, Kate)

¬∃hasChild.⊤(Jane)

Processor(Pentium4)

Pytanie:

Podaj wszystkich ludzi, którzy mają przynajmniej jedną córkę i których wszystkie znane dzieci są córkami.

*instances(Person □ ∃KhasChild.Woman □
 ∀KhasChild.Woman)*

Odpowiedź:

{Mary}

Wyjaśnienie:

Baza wiedzy „wie”, że Mary ma córkę i że wszystkie znane jej dzieci Mary to córki.



Operator K w zapytaniach

Baza wiedzy:

$$\{ \exists \text{friend}. \text{Male}(\text{SUSAN}) \}$$

Pytanie	Odpowiedź
$\exists \text{friend}. \text{Male}$	{SUSAN}
$\exists \text{friend}. \mathbf{K} \text{Male}$	<пусто>
$\exists \mathbf{K} \text{friend}. \text{Male}$	<пусто>
$\mathbf{K} \exists \text{friend}. \text{Male}$	{SUSAN}



Operator K w zapytaniach (2)

Baza wiedzy:

{ \exists friend.Male(SUSAN),
friend (SUSAN, BOB),
Male(BOB) }

Pytanie	Odpowiedź
\exists friend.Male	{SUSAN}
\exists friend.KMale	{SUSAN}
\exists Kfriend.Male	{SUSAN}
K \exists friend.Male	{SUSAN}

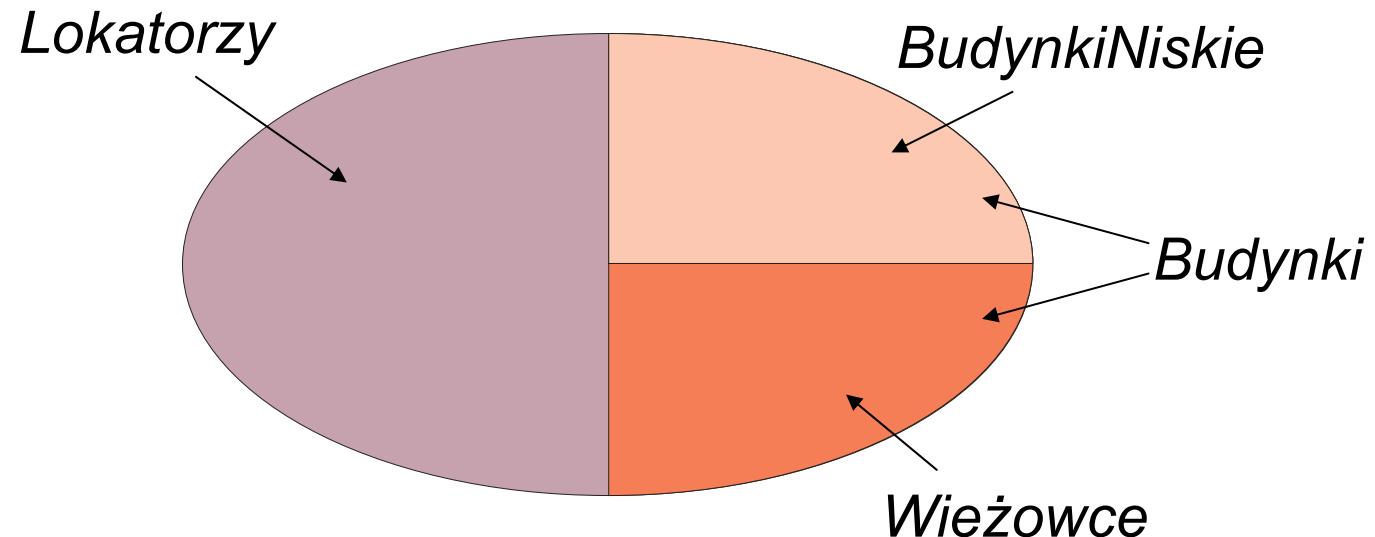


Specyfika logiki opisowej

- Logika opisowa bazuje na założeniu świata otwartego i jest ściśle powiązana z logiką predykatów pierwszego rzędu, stanowiąc jej fragment.
- Konsekwencją tych faktów są znaczące różnice w stosunku do podejścia relacyjnych i obiektowych oraz konieczność zachowania szczególnej ostrożności w stosowaniu niektórych konstrukcji.

Pułapka dziedziczenia i negacji

- Zakładamy, że w naszym świecie występują wyłącznie *Lokatorzy* i *Budynki*.
- Budynki dzielą się na *Wieżowce* i *BudynkiNiskie*; poza nimi nie ma innych budynków.





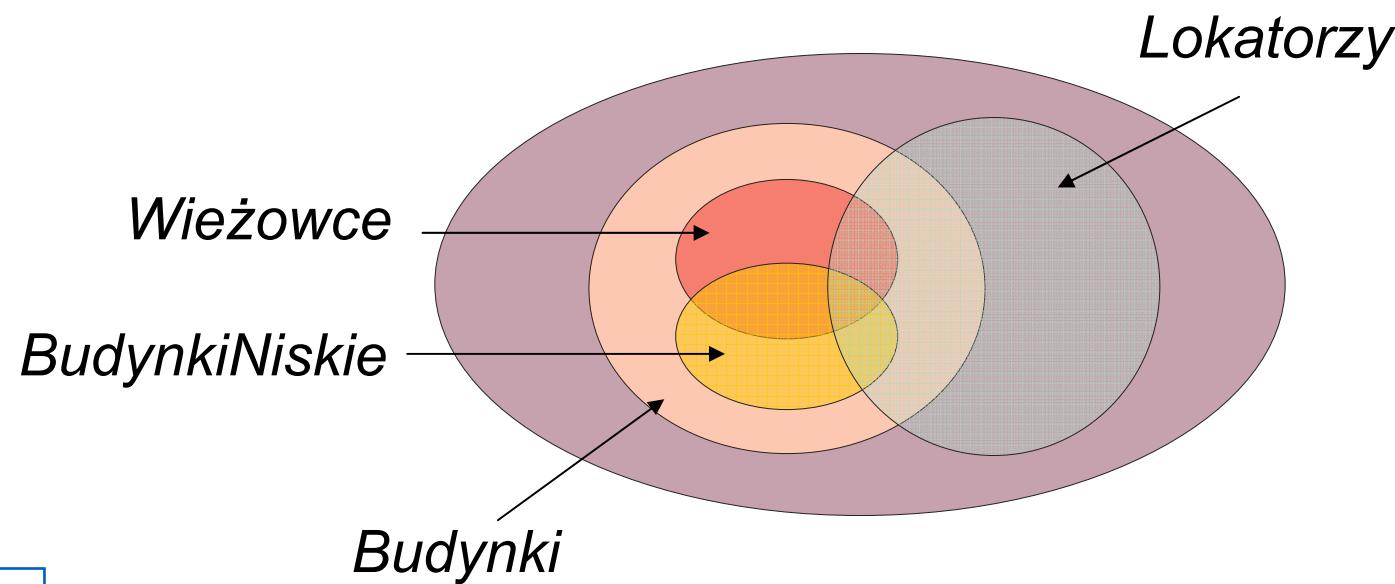
Pułapka dziedziczenia i negacji

Lokator $\sqsubseteq T$

Budynek $\sqsubseteq T$

Wieżowiec \sqsubseteq Budynek

BudynekNiski \sqsubseteq Budynek





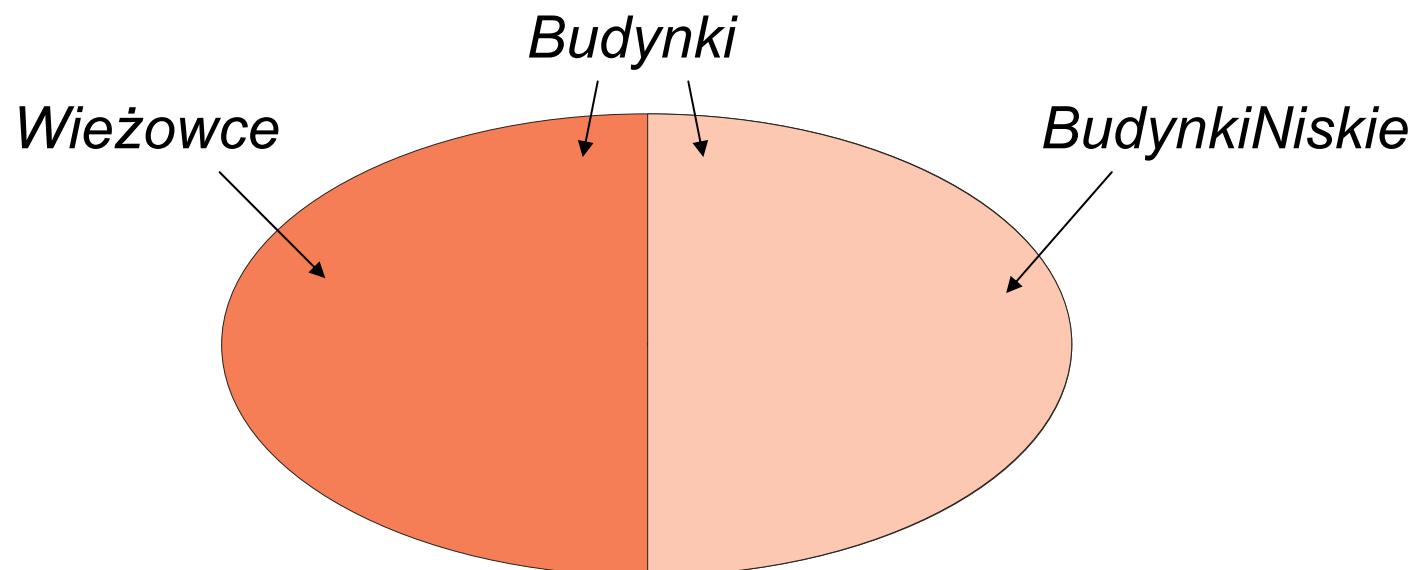
Pułapka dziedziczenia i negacji

Lokator $\equiv \neg$ Budynek

Wieżowiec \sqsubseteq Budynek

BudynekNiski \sqsubseteq Budynek

BudynekNiski $\equiv \neg$ Wieżowiec



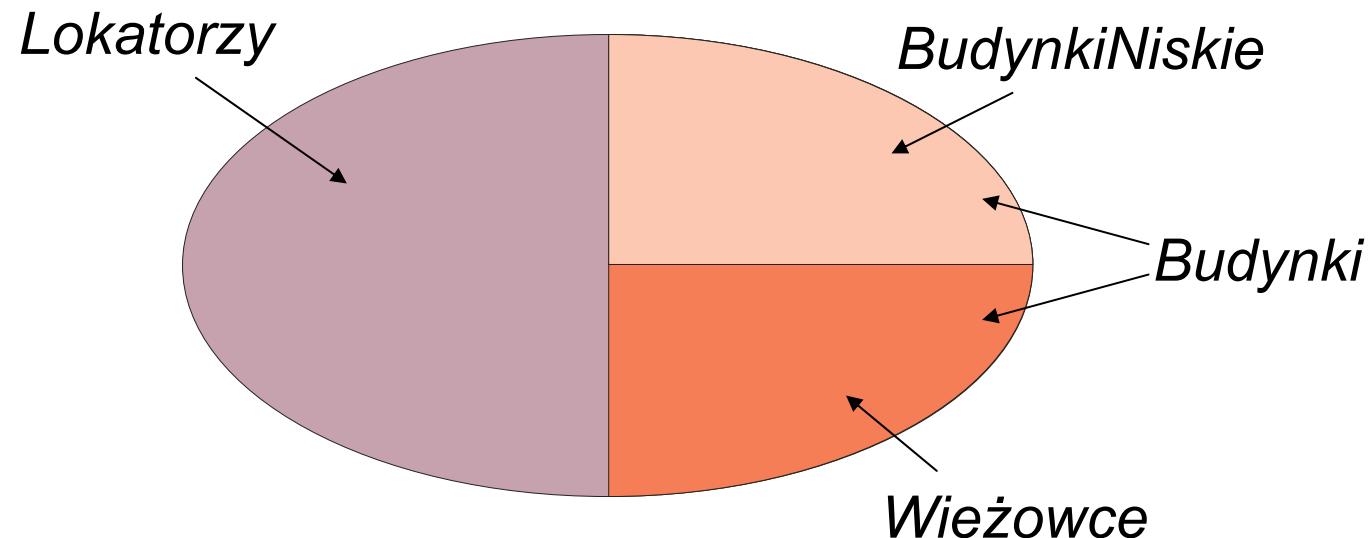


Pułapka dziedziczenia i negacji

Lokator $\equiv \neg$ Budynek

Wieżowiec \sqsubseteq Budynek

BudynekNiski \equiv Budynek $\sqcap \neg$ Wieżowiec





Określenie zakresu roli

~~$\exists \text{maCórke}. \top \equiv \exists \text{maCórke}. \text{Kobieta}$~~

Mężczyzna(Karol)
Kobieta(Anna)
 $\text{maCórke}(\text{Jan}, \text{Anna})$
 $\text{maCórke}(\text{Jan}, \text{Karol})$

Poprawne rozwiązanie:

$\exists \text{maCórke}. \neg \text{Kobieta} \equiv \perp$



Dziedziczenie i równoważność ról

$R \sqsubseteq S$

W tym rozszerzeniu aksjomat podrzędności może być stosowany dla ról. Powyższa konstrukcja oznacza, że każda para osobników będąca wystąpieniem roli R jest również wystąpieniem roli S .

Przykład:

maCiotkę \sqsubseteq maKrewnego

Wszystkie wystąpienia roli *maCiotkę* są jednocześnie wystąpieniami roli *maKrewnego*.



Role odwrotne i domknięcie przechodnie

R^-

Rola oznaczająca rolę odwrotną w stosunku do R , tj. ilekroć para osobników (a, b) jest wystąpieniem roli R , to para (b, a) jest wystąpieniem roli R^- .

Przykłady:

$$\text{maMęża} \equiv \text{maŻoneż}^-$$

R^*

Rola oznaczająca domknięcie przechodnie roli R , tj. osoby, między którymi przebiega dowolnie długi łańcuch roli R , są ze sobą w relacji R^* .

Przykłady:

$$\text{maPrzodka} \equiv \text{maOjca}^*$$

$$\text{maPrzodka} \equiv \text{maPrzodka}^*$$



Logika opisowa a OWL

- OWL-DL opiera się na ekspresywnym dialekcie logiki opisowej zwanym *SHIΩN*.
- Wszystkie konstrukcje OWL mają swoje odpowiedniki w tym dialekcie logiki opisowej.
- OWL różni się od logiki opisowej stosowanym słownictwem – często te same byty określane są różnymi terminami.



Konstrukcje OWL (1)

- Właściwości
 - tranzytywne,
 - symetryczne,
 - odwrotne,
 - funkcjonalne,
 - odwrotne funkcjonalne
- Zakresy właściwości
 - obiekty (ObjectProperty)
 - wartości (DataProperty)
- Dziedziny właściwości
 - tylko obiekty
- Cecha obiektu (*feature*): właściwość funkcjonalna o zakresie typu ObjectProperty
- Atrybut obiektu (*attribute*): właściwość funkcjonalna o zakresie typu DataProperty.



Konstrukcje OWL (2)

- Dziedziczenie (zawieranie się) klas
- Dziedziczenie (zawieranie się) właściwości
- Operatory zbiorowe na klasach:
 - przecięcie
 - unia
 - dopełnienie
- Ograniczenia liczebnościowe właściwości
- Dziedziny konkretne (liczby,łańcuchy,...)
- Przestrzenie nazewnicze



Ekspresywne logiki opisowe

- Logika opisowa to rodzina formalizmów – logiki opisowe różnią się zakresem wykorzystywanych konstruktorów,
- Ekspresywne konstruktory (wykraczające poza \mathcal{ALC}) obejmują m.in.:
 - ograniczenia liczebnościowe,
 - dziedziczenie ról,
 - role odwrotne i domknięcia przechodnie,
 - koncepty wyliczane,
 - ...



Słownictwo DL a OWL

OWL	DL
klasa	koncept
obiekt	osobnik
właściwość	rola
Datatype	dziedzina konkretna
dziedzina właściwości	(dziedzina roli)
zakres właściwości	(zakres roli)



Wzajemna odpowiedniość DL a OWL (konstruktory)

OWL	DL
intersectionOf	$C_1 \sqcap C_2 \sqcap \dots \sqcap C_n$
unionOf	$C_1 \sqcup C_2 \sqcup \dots \sqcup C_n$
complementOf	$\neg C$
someValuesFrom	$\exists R.C$
allValuesFrom	$\forall R.C$



Wzajemna odpowiedniość DL a OWL (konstruktory)

OWL	DL
minCardinality	$\geq n R. \top$
maxCardinality	$\leq n R. \top$
inverseOf	R^-
oneOf	$\{a_1, a_2, \dots, a_n\}$
hasValue	$\exists R. \{a\}$



Wzajemna odpowiedniość DL a OWL (aksjomaty)

OWL	DL
subClassOf	$C_1 \sqsubseteq C_2$
equivalentClass	$C_1 \equiv C_2$
disjointWith	$C_1 \sqsubseteq \neg C_2$
subPropertyOf	$R_1 \sqsubseteq R_2$
equivalentProperty	$R_1 \equiv R_2$



Wzajemna odpowiedniość DL a OWL (aksjomaty)

OWL	DL
sameAs	$\{x_1\} \equiv \{x_2\}$
differentFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$
FunctionalProperty	$T \sqsubseteq \leqslant 1 R. T$
InverseFunctionalProperty	$T \sqsubseteq \leqslant 1 R^{-}. T$
SymmetricProperty	$R \equiv R^{-}$
TransitiveProperty	$R \equiv R^{*}$



RDF/OWL a DIG(UT)

- RDF/OWL nie jest językiem opracowanym do formułowania zapytań
- Do zadawania zapytań istnieją specjalne języki
- Przykładami takich języków są:
 - DIG
 - DIGUT



Podstawowe konstruktory DIG(UT)- a

Nazwa konceptu	Zapis DL	Zapis DIG(UT)
Koncept uniwersalny	T	<top/>
Koncept pusty	⊥	<bottom/>
Koncept nazwany	C	<catom name="nazwa"/>
Przecięcie konceptów	$C \sqcap D$	<and> Koncept C Koncept D </and>
Suma konceptów	$C \sqcup D$	<or> Koncept C Koncept D </or>
Dopełnienie konceptu	$\neg C$	<not> Koncept C </not>



Podstawowe konstruktory DIG(UT)-a (2)

Nazwa konceptu	Zapis DL	Zapis DIG(UT)
Kwantyfikacja egzystencjalna	$\exists R.C$	<some> Rola R Koncept C </some>
Kwantyfikacja ogólna	$\forall R.C$	<all> Rola R Koncept C </all>



Zapytania DIG

- Zapytania w DIG zadaje się blokowo, tj. każde zapytanie składa się z podzapytań:

```
<asks>
  <podzapytanie1 id="q1">
  <podzapytanie2 id="q2">
  ...
</asks>
```
- Podzapytania mogą dotyczyć terminologii i/lub opisu świata.



Przykłady podzapytań DIG (1)

- Zapytanie o subsumcję konceptów:

```
<subsumes id="id">  
    koncept 1  
    koncept 2  
</subsumes>
```

```
<subsumes id="q1">  
    <some>  
        <ratom name="maDziecko"/>  
        <top/>  
    </some>  
    <catom name="Rodzic"/>  
</subsumes>
```



Przykłady podzapytań DIG (2)

- Zapytanie o przynależność osobnika do konceptu:

```
<instance id="id">
    osobnik
    koncept
</subsumes>

<instance id="q1">
    <individual name="Anna"/>
    <catom name="Matka"/>
</subsumes>
```

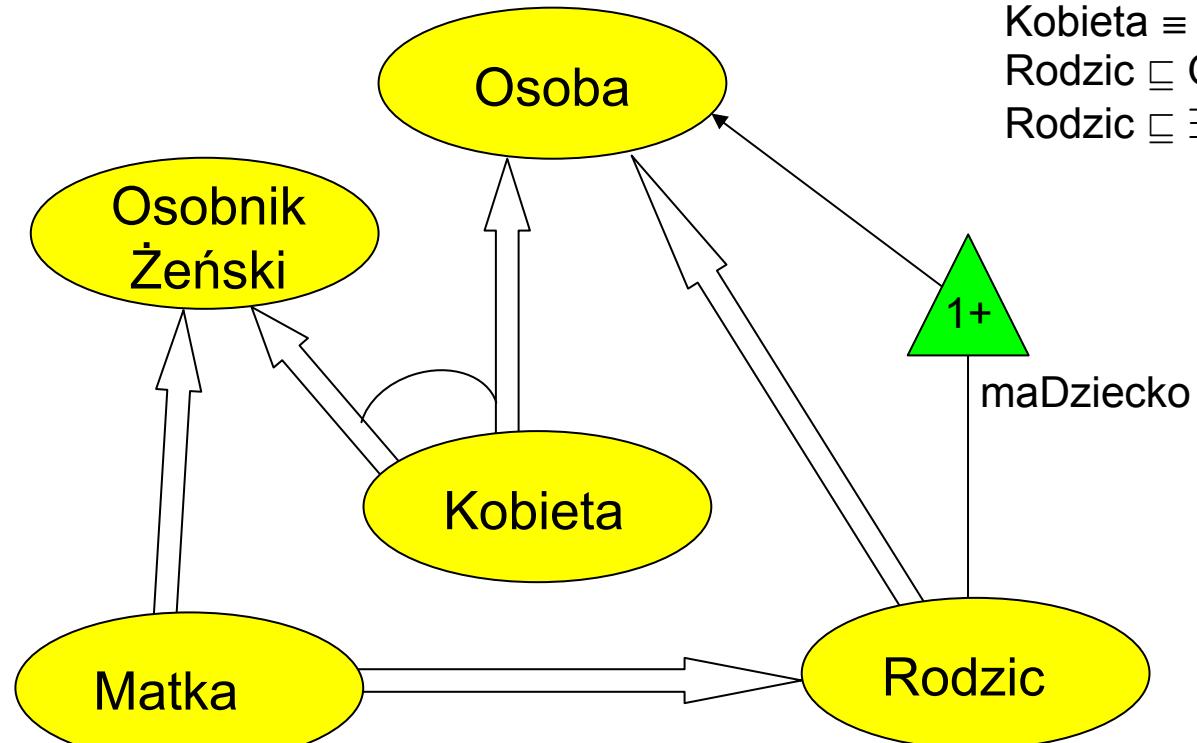


Inne podzapytania DIG

- Terminologiczne:
 - równoważność konceptów,
 - rozłączność konceptów,
 - spełnialność,
 - przodków,
 - potomków.
- Dotyczące osobników:
 - wszystkie koncepty, którymi można opisać danego osobnika,
 - listę wszystkich wystąpień konceptu.

<http://km.pg.gda.pl/kmg/digut.html>

Sieć semantyczna a wnioskowanie

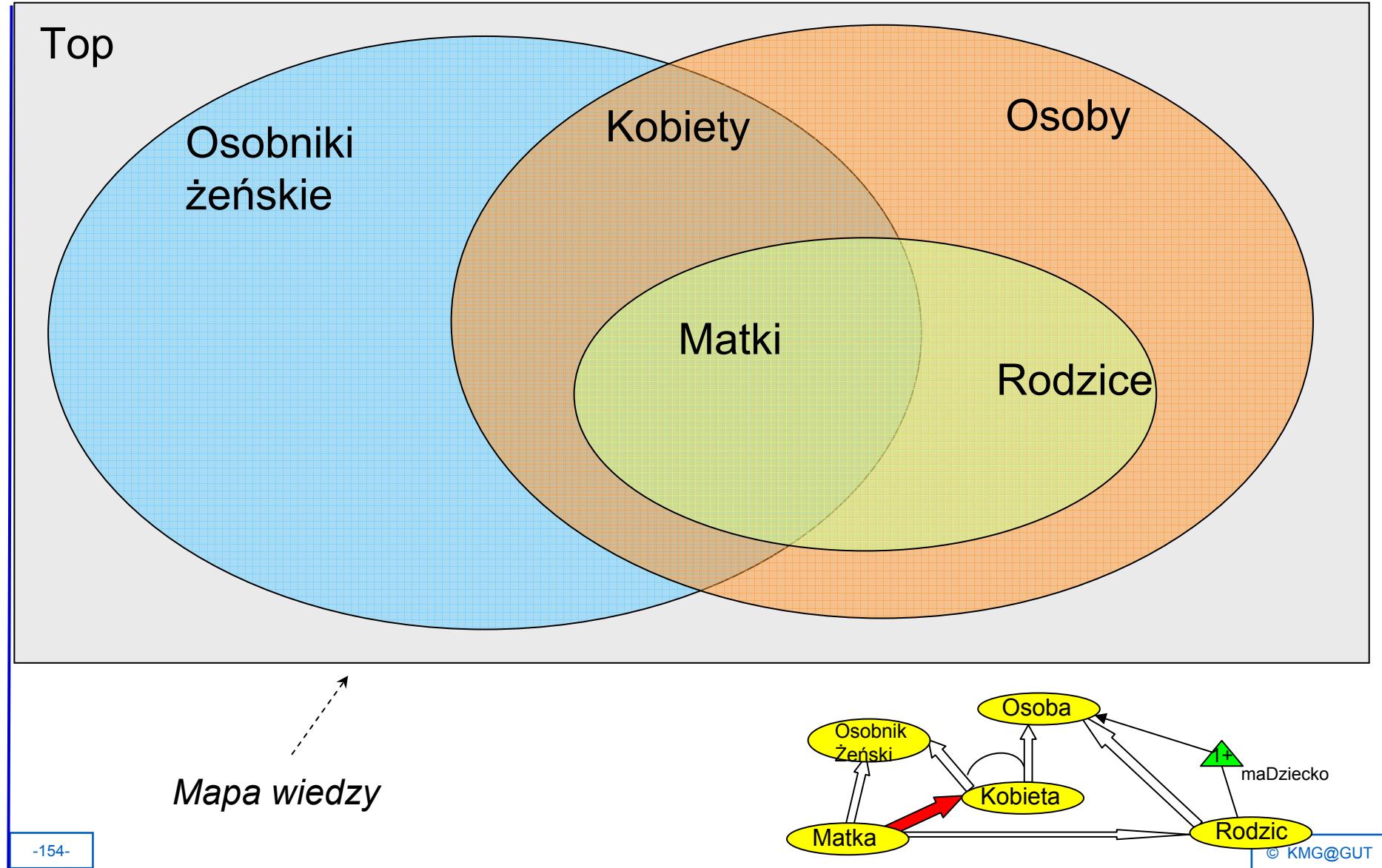


Matka ⊑ OsobnikŻeński
Matka ⊑ Rodzic

Kobieta ≡ OsobnikŻeński ∩ Osoba
Rodzic ⊑ Osoba
Rodzic ⊑ $\exists \text{maDziecko} . \text{Osoba}$

Jaki jest związek pomiędzy Matką a Kobietą?

Przykład wnioskowania





Spis treści...

1. Wprowadzenie do ontologii
 2. Wczesne ontologiczne metody reprezentacji wiedzy
 3. Semantyczny Internet (Semantic Web)
 4. Resource Description Framework (RDF)
 5. Web Ontology Language (OWL)
 6. Logika opisowa (DL) jako fundament ontologicznych metod reprezentacji wiedzy
- 7. Elementy inżynierii ontologii**
8. Źródła





Ontologia w Semantic Web

Ontologia jest **formalną**, **jawną** specyfikacją **wspólnej** **koncepcjalizacji**.

formalna

czytelna dla komputera

jawną

rodzaje użytych konceptów i ograniczenia na ich używanie są jawnie wyspecyfikowane

wspólna

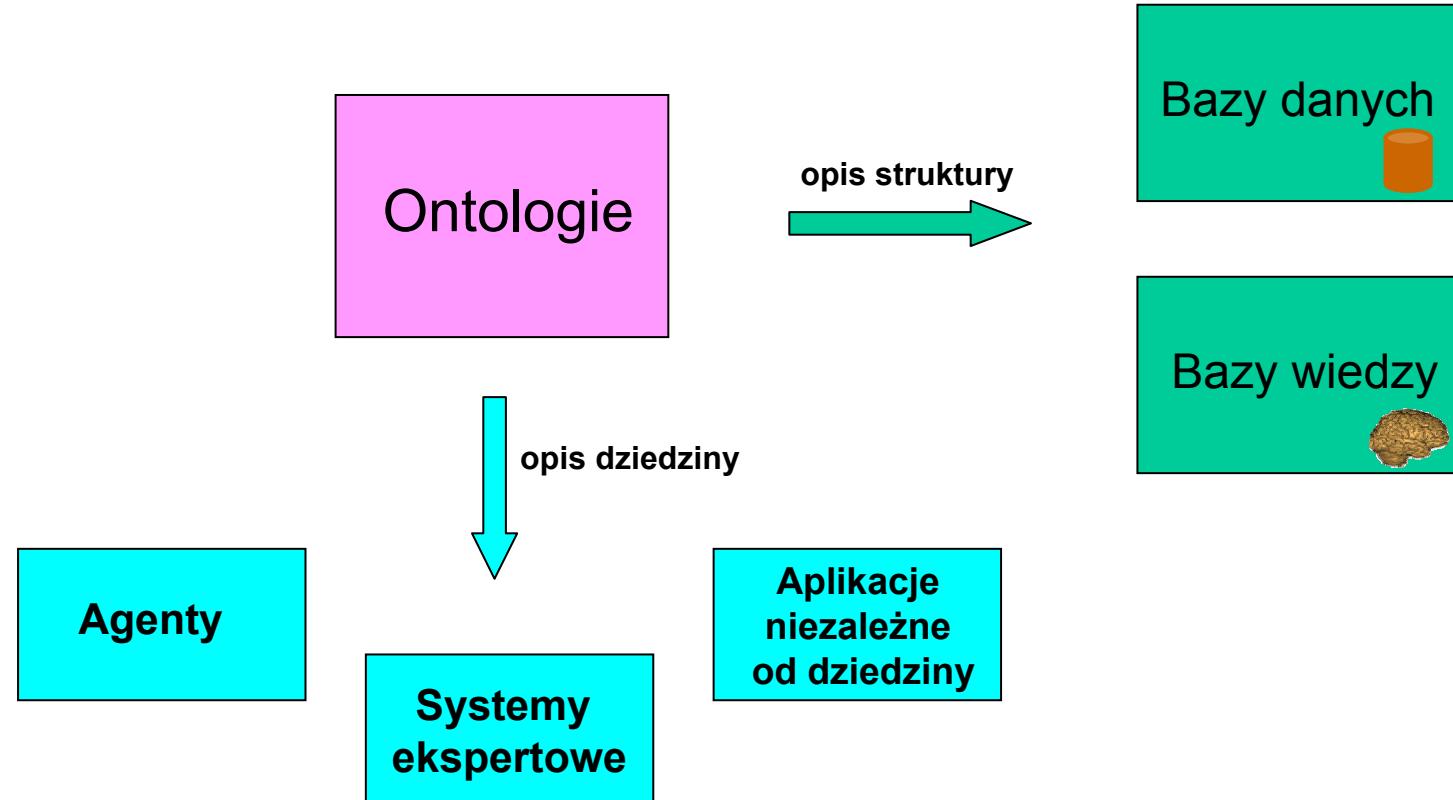
ontologia obejmuje wiedzę ogólnie uznawaną, tzn. nie „prywatną”, ale akceptowaną przez grupę ludzi

koncepcjalizacja

abstrakcyjny model pewnego realnego zjawiska, który identyfikuje istotne koncepty tego zjawiska

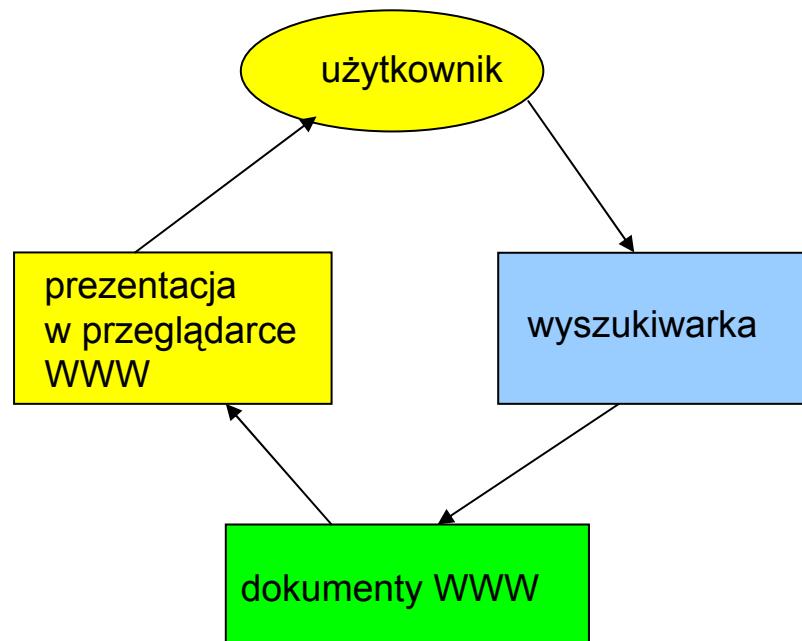


Rola ontologii w Semantic Web

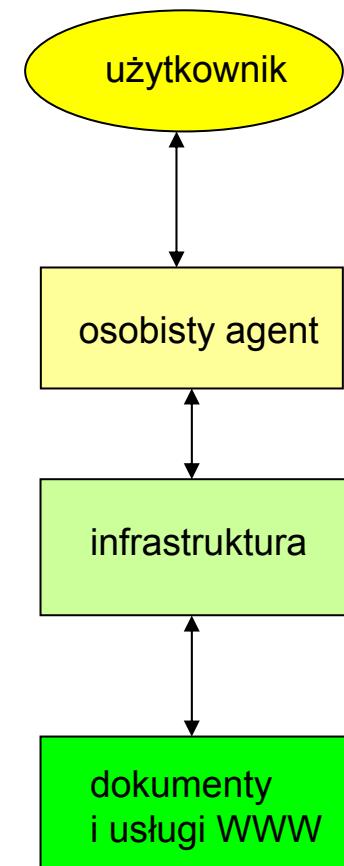


Rola agentów

w klasycznym Internecie



w Semantic Web





Semantic Web a WWW

- Semantic Web nie ma zastąpić sieci WWW - jest i będzie rozwijana niezależnie.
- Semantic Web to nie kolejna scentralizowana baza wiedzy – jest to raczej zbiór (częściowo połączonych ze sobą) ontologii i baz wiedzy.
- Głównym celem jest rozwój technik umożliwiających **współużytkowanie** (ang. *sharing*) wiedzy zawartej w licznych, rozrzuconych po Internecie, dynamicznie zmieniających się źródłach.
- Ważnym środkiem do osiągnięcia tego celu jest **ponowne użycie** (ang. *reuse*) istniejących komponentów związanych z wiedzą do budowy nowych aplikacji opartych na wiedzy.

Korzyści? Oszczędność pieniędzy, czasu i zasobów.



Zadania inżynierii ontologii

- Użycie ontologii podczas budowania nowej ontologii z wykorzystaniem już istniejących ontologii – **integracja ontologii** (*ontology integration*).
- Użycie ontologii poprzez łączenie różnych ontologii dotyczących tego samego tematu w jedną ontologię ujednolicającą wszystkie integrowane ontologie – **scalanie ontologii** (*ontology merging*).
- Użycie ontologii w konkretnych zastosowaniach – **używanie ontologii** (*ontology using*).
- Tworzenie nowych ontologii – **budowanie ontologii** (*ontology development*).



Rola słowników

WordNet

- Słownik leksykalny języka angielskiego.
- Wyrazy są łączone w grupy synonimiczne (grupy słów o tym samym znaczeniu).
- Zawiera powiązania pomiędzy różnymi formami tych samych słów.

Na podstawie WordNet możliwe jest tworzenie różnych funkcji miar podobieństwa z wykorzystaniem zależności pomiędzy słowami (np. czy dwa słowa należą do jednego zbioru synonimów).

Umożliwia to dopasowywanie do siebie różnych ontologii
(ontology alignment)



Budowanie nowych ontologii

- Jeśli nie istnieją ontologie, które mogą zostać wykorzystane, należy budować nowe ontologie.
- Istnieją **edytory ontologii**.
- Język OWL jest standardem W3C wspierającym budowanie i sformalizowanie ontologii.

http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html



Przykładowe dostępne ontologie

Ontologie dziedzinowe:

- PhySys – fizyka
- EngMath – matematyka
- KACTUS ontology library – biblioteka podstawowych ontologii technicznych
- ARPA/Rome Laboratory Planning Initiative – ontologia wspierająca planowanie
- GALEN – ontologia medyczna
- UMLS – ontologia medyczna

Ontologie języka naturalnego:

- SENSUS – ontologia języka naturalnego

Ontologie wysokiego poziomu (*upper ontology*):

- Dublin Core



Dostępne ontologie - adresy

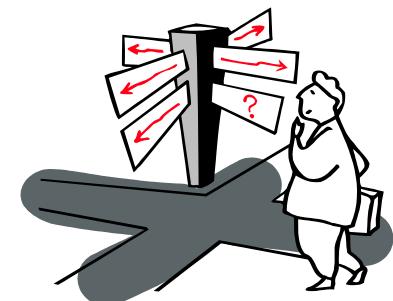
- <http://www.daml.org/ontologies/>
- <http://hcs.science.uva.nl/projects/Kactus/toolkit/intro.html>
- <http://www.isi.edu/natural-language/resources/sensus.html>
- <http://aaaipress.org/Library/ARPI/arp96contents.php>
- http://www.openclinical.org/prj_galen.html
- <http://www.nlm.nih.gov/research/umls/>
- <http://wordnet.princeton.edu/>
- <http://dublincore.org/>
- ...



Spis treści...

1. Wprowadzenie do ontologii
2. Wczesne ontologiczne metody reprezentacji wiedzy
3. Semantyczny Internet (Semantic Web)
4. Resource Description Framework (RDF)
5. Web Ontology Language (OWL)
6. Logika opisowa (DL) jako fundament ontologicznych metod reprezentacji wiedzy
7. Elementy inżynierii ontologii

8. Źródła





Źródła

Sieć Semantyczna (Semantic Web)

- <http://www.semanticweb.org/>
- <http://www.ics.forth.gr/isl/swprimer/>

Języki reprezentacji wiedzy RDF, OWL

- <http://www.w3.org/RDF/>
- <http://www.w3.org/2004/OWL/>
- <http://km.pj.gda.pl/kmg/digut/>

Logika opisowa

- Baader F. A., McGuiness D. L., Nardi D., Patel-Schneider P. F.: *The Description Logic Handbook: Theory, implementation, and applications*, Cambridge University Press, 2003.
- Staab, S., Studer, R. (eds). *Handbook on Ontologies*. Springer Verlag, 2004.
- <http://dl.kr.org/>



Źródła

Narzędzia do inżynierii wiedzy

- <http://protege.stanford.edu/>
- <http://www.cs.man.ac.uk/~horrocks/FaCT/>
- <http://www.sts.tu-harburg.de/~r.f.moeller/racer/>
- <http://jena.sourceforge.net/inference/index.html>
- <http://instancesstore.man.ac.uk/>

Tworzenie ontologii

- Staab, S., Studer, R. (eds). *Handbook on Ontologies*. Springer Verlag, 2004.
- http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html



Źródła

Istniejące ontologie

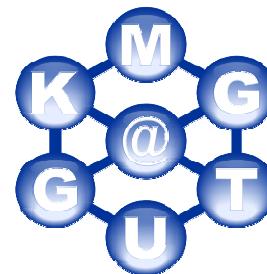
- Borst, P.: Construction of Engineering Ontologies for Knowledge Sharing and Reuse, PhD thesis, Twente University, 1997.
- Gruber, T. & Olsen, G. R. An Ontology for Engineering Mathematics, in E. S. J. Doyle & P. Torasso, eds, 'KR94 Proceedings', Morgan Kaufmann, pp. 258–269, 1994.
- <http://hcs.science.uva.nl/projects/Kactus/toolkit/intro.html>
- <http://www.isi.edu/natural-language/resources/sensus.html>
- <http://aaaipress.org/Library/ARPI/arp196contents.php>
- http://www.openclinical.org/prj_galen.html
- <http://www.nlm.nih.gov/research/umls/>
- <http://wordnet.princeton.edu/>
- <http://dublincore.org/>



Źródła

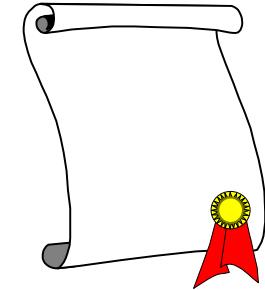
Różne zagadnienia
związane z ontologiami i zarządzaniem wiedzą

<http://km.pg.gda.pl/kmg/>





Źródła



Książki:

- Baader F. A., McGuiness D. L., Nardi D., Patel-Schneider P. F.: *The Description Logic Handbook: Theory, implementation, and applications*, Cambridge University Press, 2003.
- Staab, S., Studer, R. (eds). *Handbook on Ontologies*. Springer Verlag, 2004.

Źródła internetowe:

- www.w3.org/XML
- www.w3.org/RDF
- www.w3.org/OWL
- www.w3.org/2001/SW
- protege.stanford.edu
- www.ksl.stanford.edu
- www.daml.org
- km.pg.gda.pl



Materiały źródłowe

XML, XMLSchema

www.w3.org/XML



RDF, RDFSchema

www.w3.org/RDF



Web Ontology Language Guide

www.w3.org/OWL



Protégé

protege.stanford.edu

(dokumentacja+program+dodatki+ontologie)

Semantic Web

www.w3.org/2001/sw